CRANFIELD UNIVERSITY

D. CASANOVA

ON MINIMUM TIME VEHICLE MANOEUVRING: THE THEORETICAL OPTIMAL LAP

SCHOOL OF MECHANICAL ENGINEERING

PhD THESIS

CRANFIELD UNIVERSITY

SCHOOL OF MECHANICAL ENGINEERING

PhD THESIS

Academic Year 2000-2001

D. CASANOVA

ON MINIMUM TIME VEHICLE MANOEUVRING: THE THEORETICAL OPTIMAL LAP

Supervisor: Prof. Robin S. Sharp

November 2000

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Abstract

This work is a research on the minimum time vehicle manoeuvring problem, with a particular application to finding the minimum lap time for a Formula One racing car. The proposed method allows to solve the general problem of evaluating the vehicle lateral and longitudinal controls which yield the minimum time required to traverse a lap of a circuit.

The minimum time vehicle manoeuvring problem is formulated as one of Optimal Control and is solved using mathematical programming methods. Novel techniques are employed to solve the resulting non-linear programming problem which allow to achieve effective optimisation with satisfactory accuracy, robustness and computational efficiency. Particularly, the proposed solution strategy is generally applicable to any arbitrarily complex vehicle mathematical model.

Car and circuit models are set up, and the optimisation program is applied to investigate the sensitivity of the vehicle performance with respect to vehicle design parameters, such as the yaw moment of inertia, the total mass and the weight distribution. Furthermore, the minimum time manoeuvring problem is solved for very different vehicle configurations. The optimisation program accurately quantifies the vehicle performance in terms of manoeuvre time, and the nature of the optimal solution is shown to be always in excellent agreement with the dynamic properties of the vehicle model.

A part of the work is devoted to the development of a strategy to obtain an initial estimate of the racing line and of the vehicle lateral and longitudinal controls to be used at the start of the optimisation. Two algorithms to compute the racing line using on board measured data from the real car are presented. A new mathematical model for the vehicle steering control is derived. The model uses multiple preview information of the intended path. Its structure derives from linear optimal preview control theory, but it is adapted to deal with non-linear vehicle operations arising from the inevitable tyre force saturation in vigorous manoeuvring. The excellent path following capability of the model is demonstrated by solving various path following tasks involving moderate manoeuvring and racing speeds.

Acknowledgements

I would like to express my sincere gratitude to Prof. Robin Sharp, who made this work possible in first place. His experience and his guidance have been invaluable throughout the whole project. His unlimited enthusiasm for the subject has been extremely motivating and inspirational.

This work has been carried out with the technical and financial support of Benetton Formula Ltd., Formula One team. I am particularly grateful to Mr. Pat Symonds, Head of Engineering at Benetton Formula, for his constant support and technical advice regarding the development of the work. I should also like to mention Mr. Edwin Thiron, Vehicle Dynamics engineer at Benetton Formula, for supplying technical data and information.

The application of Automatic Differentiation has been carried out with the technical support of the Numerical Optimisation Centre, University of Hertfordshire, Hatfield. I would like to thank Prof. Bruce Christianson for introducing me to the use of Automatic Differentiation and for his valuable advice. I am very grateful to Mr. Mark Final, formerly a research student at the Numerical Optimisation Centre, who developed the Automatic Differentiation algorithm which I used, and who has been constantly in touch during the development of the work.

The optimisation algorithm was supplied by Prof. Philip Gill, Department of Mathematics, University of California, San Diego. His software has greatly contributed to the success of my work.

I would like to thank all the people and friends at Cranfield University, who have contributed in making the working environment enjoyable and stimulating. Among all, I would like to mention Mr. Matteo Bettella and Mr. Jean Michel Rogero for their help and advice on matters regarding computer programming.

My final, special thanks are for my family, to whom I am pleased to dedicate my work. They supported me for all these years. We have made it!

Table of contents

Nomenclature

LIST OF SYMBOLS

a_{vx}, a_{vy}	vehicle longitudinal and lateral acceleration
A	general set of active inequality constraints
B_f, B_r	front and rear axles braking torque distribution coefficients
B_{T_MAX}	maximum braking torque applicable to wheels
c	general constraint
$\mathbf{c}$	vector of constraints
$C_{f\phi}$, $C_{r\phi}$	front and rear roll stiffness
CG	vehicle centre of mass
C_x	aerodynamic drag coefficient
d	distance of the vehicle centre of mass from a reference line
$\mathbf{d}$	vector representing a generic direction in a n-dimensional space
$\mathbf{D}$	diagonal matrix with problem scaling coefficients
e_i	i^{th} lateral off-set measured from the optical lever to the ideal path
e_r	round-off errors
e_t	truncation errors
e_ψ	vehicle attitude error
E	general set of equality constraints
E_{brk}	engine braking torque
E_x	end longitudinal gap in the reconstructed racing line
E_y	end lateral gap in the reconstructed racing line
E_ψ	end angular gap in the reconstructed racing line
$\mathbf{E_{trq}}$	engine output torque map
$\mathbf{E_{th}}$	throttle input vector to the engine map
$\mathbf{E_{rpm}}$	engine rotational velocity input vector to the engine map
F_{ax}	aerodynamic drag
F_{az}	aerodynamic lift
F_{vx}	vehicle total longitudinal thrust, used in the driver model
F_{xi}	i^{th} wheel tyre longitudinal force, in vehicle reference axes system
F_{yi}	i^{th} wheel tyre lateral force, in vehicle reference axes system
F_{xwi}	i^{th} wheel tyre longitudinal force, in wheel reference axes system
F_{ywi}	i^{th} wheel tyre lateral force, in wheel reference axes system
F_{zi}	i^{th} wheel vertical load
g	gravitational acceleration
G_{D1}	differential torque transfer off-set due to internal friction
G_{D2_O}	differential torque transfer gain in overrun
G_{D1_P}	differential torque transfer gain on power
G_R	gear ratio
h_g	height of CG
h_{rf}, h_{rr}	front and rear roll centre height
H	Hamiltonian function
$\mathbf{H}$	Hessian matrix
I	general set of inequality constraints

I_{zz}	vehicle yaw moment of inertia
$\mathbf{I}$	identity matrix
J	performance measure
J_m	engine polar moment of inertia
J_{wf}, J_{wr}	front and rear wheels polar moment of inertia
k_i	i^{th} wheel longitudinal slip
k_t	curvature of the vehicle path or of the road centre line
K_i	steer angle control gain for i^{th} lateral offset input
K_ψ	steer angle control gain for the vehicle attitude error
l_f, l_r	distance of the CG from the front and the rear axle
l_i	i^{th} preview point on the optical lever
L	Lagrangian function
L_p	preview distance
L_{SL}	maximum allowed wheel longitudinal slip, used in the driver model
m	counter for the number of state trajectory segments
M	vehicle total mass
M_{zz}	unbalanced vehicle yaw moment
n	counter for the total number of control nodes
p	counter for the number of state variables
p_1	off-set compensation parameter the in trajectory reconstruction algorithm
p_2	linear compensation parameter in the trajectory reconstruction algorithm
p_3	gain compensation parameter in the trajectory reconstruction algorithm
$\mathbf{p}$	vector of compensation parameters in the trajectory reconstruction algorithm
q	counter for the number of control variables
r_i	i^{th} preview point on the ideal path
r_t	radius of the vehicle path or of the road centre line
R_f, R_r	front and rear wheel radii
R_{sf}, R_{sr}	front and rear roll stiffness distribution
s	path length co-ordinate
s_{lim}	major step limit in the optimisation algorithm
$\mathbf{s_m}$	set of path co-ordinates for the trajectory discretisation grid
$\mathbf{s_n}$	set of path co-ordinates for the control discretisation grids
$\mathbf{s_P}$	vector of preview instances along the optical lever
$\mathbf{s_{stn}}$	steer angle communication grid
$\mathbf{s_R}$	vector of preview instances along the ideal path
$\mathbf{s_{tbn}}$	throttle/brake control communication grid
S	total path length or circuit length
Sat_i	i^{th} lateral off-set control saturation level
Sat_{LT}	global lateral off-set control saturation level
Sat_{TOT}	global saturation level for steer angle control
S_{CF}	time to travelled distance scaling factor
S_f	vehicle frontal area
t	time
t_f, t_r	final time
$\mathbf{t_n}$	time instances for discretised control history
T	total manoeuvre time
T_i	torque applied to the i^{th} wheel

T_B total braking torque
T_D torque input to differential
T_E actual engine output torque
T_p preview time
T_T magnitude of the differential torque transfer
$\mathbf{T}$ matrix of path information
u_{st} steer angle control input
u_{tb} throttle/brake control input
$\mathbf{u}$ vector of continuous control variables
$\mathbf{u}_L$ lower control boundary, continuous control history
$\mathbf{u}_U$ upper control boundary, continuous control history
$\mathbf{u_n}$ vector of all discrete control parameters
$\mathbf{u_{stn}}$ discretised steer angle control input
$\mathbf{u_{tbn}}$ discretised throttle/brake control input
$\mathbf{u}_\mathbf{n}^L$ lower control boundary, discretised control history
$\mathbf{u}_\mathbf{n}^U$ upper control boundary, discretised control history
U general set of admissible control histories, i.e. controls which do not violate constraints
V_t target velocity in path following problems
V_{vx} vehicle longitudinal velocity
V_{vy} vehicle lateral velocity
w_t road width
W_b wheelbase
W_n digital filter cutoff normalised frequency
x_t vehicle path or road centre line co-ordinates
x_v vehicle centre of mass x co-ordinate in a global reference axis system
$\mathbf{x}$ general vector of continuous system state variables; particularly, for the seven degrees of freedom vehicle model:

- x_1 yaw angle;
- x_2 yaw rate;
- x_3 longitudinal velocity;
- x_4 lateral velocity;
- x_5 x co-ordinate of the centre of gravity;
- x_6 y co-ordinate of the centre of gravity;
- x_7 front left wheel angular position;
- x_8 front left wheel angular velocity;
- x_9 front right wheel angular position;
- x_{10} front right wheel angular velocity;
- x_{11} rear left wheel angular position;
- x_{12} rear left wheel angular velocity;
- x_{13} rear right wheel angular position;
- x_{14} rear right wheel angular velocity;

$\mathbf{x_L}$ vector of the x co-ordinates of the preview points on the optical lever
$\mathbf{x_R}$ vector of the x co-ordinates of the preview points on the ideal path
$\mathbf{x_m}$ vector of discrete state parameters

X	general set of feasible state trajectories, i.e. trajectories which do not violate constraints
y_t	vehicle path or road centre line co-ordinates
y_v	vehicle centre of mass y co-ordinate in a global reference axis system
$\mathbf{y}$	vector of independent optimisation variables
$\mathbf{y}_L$	lower independent optimisation variables boundary
$\mathbf{y}_U$	upper independent optimisation variables boundary
$\mathbf{y_L}$	vector of the y co-ordinates of the preview points on the optical lever
$\mathbf{y_R}$	vector of the y co-ordinates of the preview points on the ideal path
α_i	i^{th} wheel slip angle
δ	road wheel steer angle
$\boldsymbol{\delta}$	search direction in optimisation algorithms
$\boldsymbol{\Delta}_L$	vector of the relative positions of the preview points on the optical lever
$\Delta\mathrm{T}$	actual differential torque transfer
Δ_x	end longitudinal gap compensation coefficient in the racing line reconstruction algorithm
Δ_y	end lateral gap compensation coefficient in the racing line reconstruction algorithm
Δ_ψ	end angular gap compensation coefficient in the racing line reconstruction algorithm
ε_d	fractional accuracy in derivative evaluation
ε_f	fractional accuracy in function evaluation
ϕ	merit function
γ_i	i^{th} wheel camber angle
λ	Lagrange multiplier
$\boldsymbol{\lambda}$	vector of Lagrange multipliers
ν	angular direction of the vehicle absolute velocity in a global reference axis system
ρ	air density
ω	general rotational velocity
τ	step size
ξ_i	vector of defects at the i^{th} node of the state discretisation mesh
ψ_t	vehicle path or road centre line tangent angle
ψ_v	vehicle attitude angle in a global reference axis system

MATHEMATICAL NOTATIONS

- Normal-italic letters, either upper of lower case indicate general variables, e.g. a, B, τ, or names of particular functions, e.g. L for Lagrangian function;
- Bold letters, lower case indicate a vector, e.g. $\mathbf{a}$, ξ;
- Bold letters, upper case indicate a matrix, e.g. $\mathbf{H}$.
- The following symbols are used with the corresponding meaning:

$\in$	reads: "belongs to"
$\forall$	reads: "for any"
$a \equiv b$	reads: "a coincides with b"
$a \div b$	reads: "from a to b"
$A \subseteq B$	reads: "A is a sub-set of B"
$C = A \cup B$	reads: "the set C is the union of the sets A and B"
o	reads: "little oh". For example:

$$f(x) = o(g(x)) \qquad x \to 0$$

reads: "$f(x)$ is little oh of $g(x)$ when x tends to 0". The meaning is the following:

$$\lim_{x \to 0} \left(\frac{f(x)}{g(x)} \right) = 0$$

- Gradient, Jacobian and Hessian notations:

Let $\mathbf{x}$ be a n-dimensional vector. Let also $a(\mathbf{x})$ be a scalar function and $\mathbf{b}(\mathbf{x})$ a m-dimensional vector function. The gradient of a with respect to $\mathbf{x}$ is a row vector and it reads:

$$\frac{\partial a}{\partial \mathbf{x}} = \left[\frac{\partial a}{\partial x_1} \quad \frac{\partial a}{\partial x_2} \quad \cdots \quad \frac{\partial a}{\partial x_n} \right]$$

The Jacobian of $\mathbf{b}$ with respect to $\mathbf{x}$ reads:

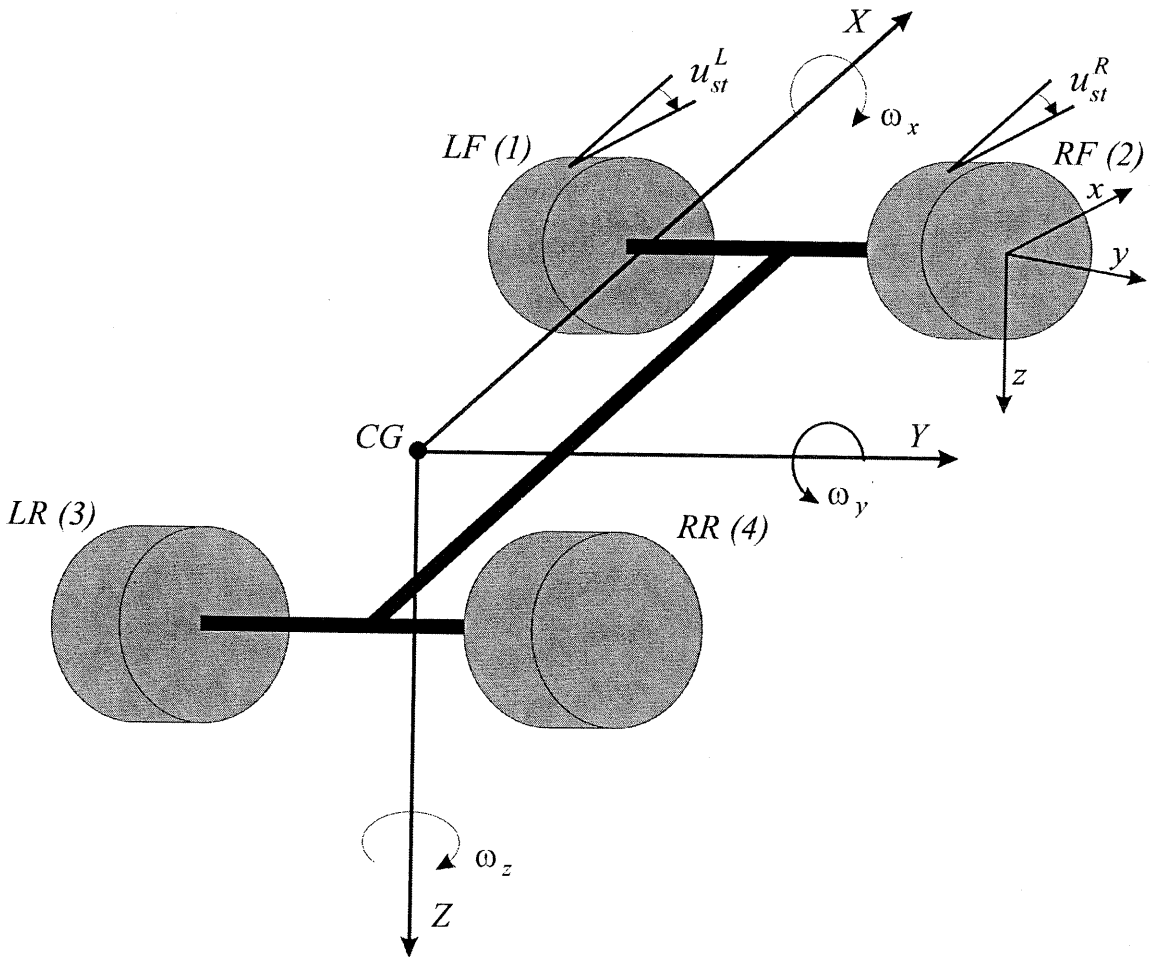
$$\frac{\partial \mathbf{b}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial b_1}{\partial x_1} & \frac{\partial b_1}{\partial x_2} & \cdots & \frac{\partial b_1}{\partial x_n} \\ \frac{\partial b_2}{\partial x_1} & \frac{\partial b_2}{\partial x_2} & & \\ \vdots & & \ddots & \\ \frac{\partial b_m}{\partial x_1} & & & \frac{\partial b_m}{\partial x_n} \end{bmatrix}$$

Finally, the Hessian matrix of a with respect to $\mathbf{x}$ reads:

$$\frac{\partial^2 a}{\partial \mathbf{x}^2} = \begin{bmatrix} \frac{\partial^2 a}{\partial x_1^2} & \frac{\partial^2 a}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 a}{\partial x_1 \partial x_n} \\ \frac{\partial^2 a}{\partial x_2 \partial x_1} & \frac{\partial^2 a}{\partial x_2^2} & & \\ \vdots & & \ddots & \\ \frac{\partial^2 a}{\partial x_n \partial x_1} & & & \frac{\partial^2 a}{\partial x_n^2} \end{bmatrix}$$

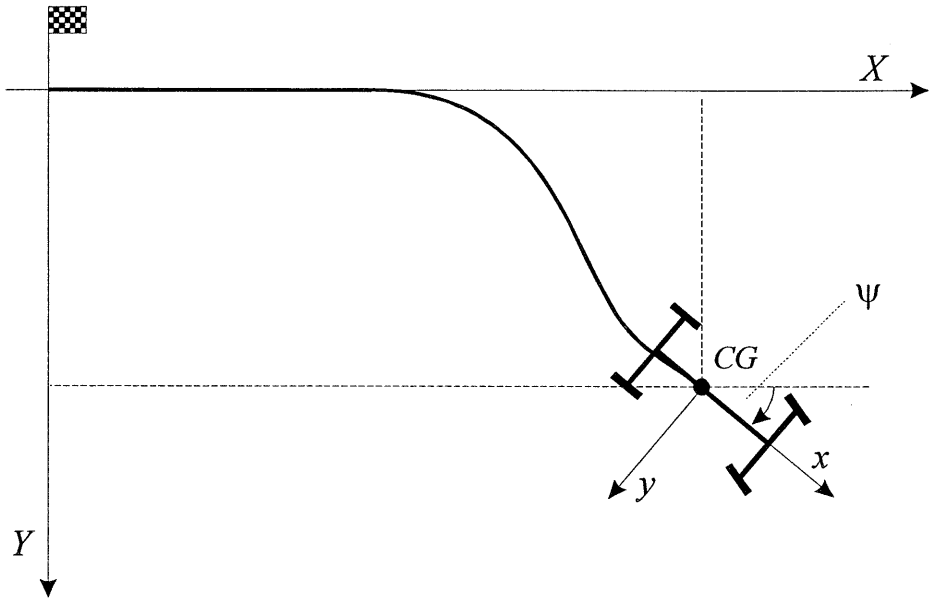
SIGN CONVENTIONS

The sign conventions used in this work are chosen according to the SAE reference axes system for a vehicle. The following figure shows the vehicle axes located at its centre of gravity, and the local axes system for a wheel. Variables which refer to one of the road wheels are indicated either with the subscripts LF, RF, LR, RR, or 1,2,3,4.



Vehicle reference axes system.

The global reference axes system fixed in space is usually located at the start of the vehicle trajectory. Vertical vehicle displacements, e.g. road elevation and camber, are not included. The following figure shows the vehicle position in the global reference axes system.



The global reference axes system.

Chapter 1.

Introduction

The essential purpose of a circuit racing car is to traverse laps of the circuit in minimum time. Mathematical models have been developed in order to predict the maximum performance as a function of vehicle parameter changes. It appears likely that the first simulation of racing car laps was by Mercedes Benz as early as the 1937-39 period. Simulation was regularly used in the mid 50s, as we may learn from the following quote by Stirling Moss:

> *...and then behind all this there was, of course, a race analysis department. If you should wonder what on Earth this might be, or do, let me say that every course was mathematically dissected and the speed at every point calculated together with the required gear in every section. From this it was child's play to tell the driver what he should do, where he should do it, and what sort of lap speeds he should achieve. So if you started by saying that on a certain corner you were coming out at 5,500 r.p.m. and you would prefer 5,300 r.p.m., in next to no time the gear ratio would be changed so that without altering your road speed this is how it would be. But then somebody else would say: "My dear chap, you're going too slow there, on that ratio you really ought to be coming out at 5,450." And, my God, when you took it seriously out of it you came at 5,450. Or perhaps 5,500 if you wished to prove that slide rules are not always infallible.*
>
> (Moss and Pomeroy, 1963)

The simple manual procedure consisted in breaking the lap into straights and constant radius corners. Firstly, the limiting velocities at each turn were found by knowing the maximum lateral performance of the car in steady-state cornering conditions. Then, the maximum accelerating and braking capabilities of the car, limited by the longitudinal tyre friction and the maximum engine and brake power, were used to calculate the velocity to go from one corner to another. In those days the problem was relatively simple, since the absence of aerodynamic downforce made the vehicle acceleration limit performance essentially independent of speed. Nowadays, circuit race cars rely on aerodynamic loads in order to achieve greater cornering, braking and driving capabilities and this makes their performance dependent upon the velocity in a more complex way. In the early days of motor racing, though, the only instrumentation available to race engineers was essentially the stop watch and the slide rule. In recent years computer technology has had a massive impact on motor sport. Firstly pioneered

by Renault in 1985[1], on board data acquisition technology has expanded onto modern racing cars to become the primary tool for vehicle performance analysis. Together with experimental data, simulation techniques are essential to aid understanding of the performance limits and to make quantitative predictions based on computer models.

In this section a short review of the simulation methodologies which are commonly used to analyse circuit race car performance is presented. For many years the underlying concept has been the assumption that a steady-state analysis is a good approximation of the actual behaviour of the car when estimating lap times. This idea is discussed and its limitations are highlighted through the qualitative investigation of the nature of minimum time vehicle manoeuvring. Then, a more realistic formulation of the problem is obtained using Optimal Control theory. A review of the relevant previous works on the subject is presented and the basis for this research is established.

Vehicle dynamics simulations for the assessment of performance and handling qualities often involve following a prescribed path. For example, it is a common practice among racing teams to use the on board measured data to reconstruct the racing line and then "run" the computer model of the vehicle along this trajectory at racing speed. Furthermore, as we shall see later on, the solution of the theoretical optimal lap problem requires a good estimate of the initial vehicle control history along the track, which, in turn, may be defined as a path following task. For this purpose a part of this research was dedicated to the development of a driver model. Hence, later in this section a short insight into the field of driver modelling is given and the fundamental theory which is applied in this work highlighted. Finally, a brief description of the techniques used for circuit modelling and race line reconstruction is also given.

1.1. STEADY-STATE PERFORMANCE ANALYSIS

Circuit racing driving is known to be very smooth and progressive. This idea led to the assumption that minimum time manoeuvring around a race course may be approximated by using the steady-state performance limits of a race car. Strictly, steady-state vehicle manoeuvring involves linear motion or cornering at constant speed. In the first case no acceleration is applied to the vehicle, while in the latter case a constant lateral acceleration is applied at the vehicle centre of gravity. These two kinematic conditions are truly steady-state, as they may persist for indefinite time. However, we may define two other conditions of motion which may be referred to as "momentary steady-state" or "quasi steady-state". These are linear motion or cornering at constant speed and constant longitudinal acceleration. In the latter case it is also assumed that the yaw rate is constant. Therefore no net moment about the yaw axis is applied to the vehicle and the tyre lateral forces are fully used to produce the lateral acceleration. This condition is also referred to as "trimmed steady-state". Quasi steady-state conditions of motion can not exist for a finite amount of time. However, it is regarded as acceptable to approximate the transient motion of a vehicle with a sequence of quasi steady-state conditions when the rate of change of the forces applied to it is sufficiently slow compared to the system dynamics. A typical example is when a car is progressively accelerating along a straight line, while a true transient would occur if the driver suddenly hits the brakes.

The most common representation of the vehicle steady-state performance

[1] Source: Autosport magazine, Vol. 153, N. 4, October 1998, p. 59.

boundaries is the *"g/g/speed"* surface, which is the envelope of a set of "*g/g*" planar diagrams representing each of the maximum longitudinal and lateral accelerations in steady-state conditions at a certain velocity (Milliken and Milliken, 1995). Figure 1.1 is a qualitative representation of the g-g diagram for a modern circuit race car with aerodynamic downforce. The performance boundaries of the vehicle in braking and cornering constantly expand when increasing the speed, as a result of the greater vertical loads on the tyres. Also the accelerating capability of the car increases, but only until the driving torque available becomes the limiting factor.

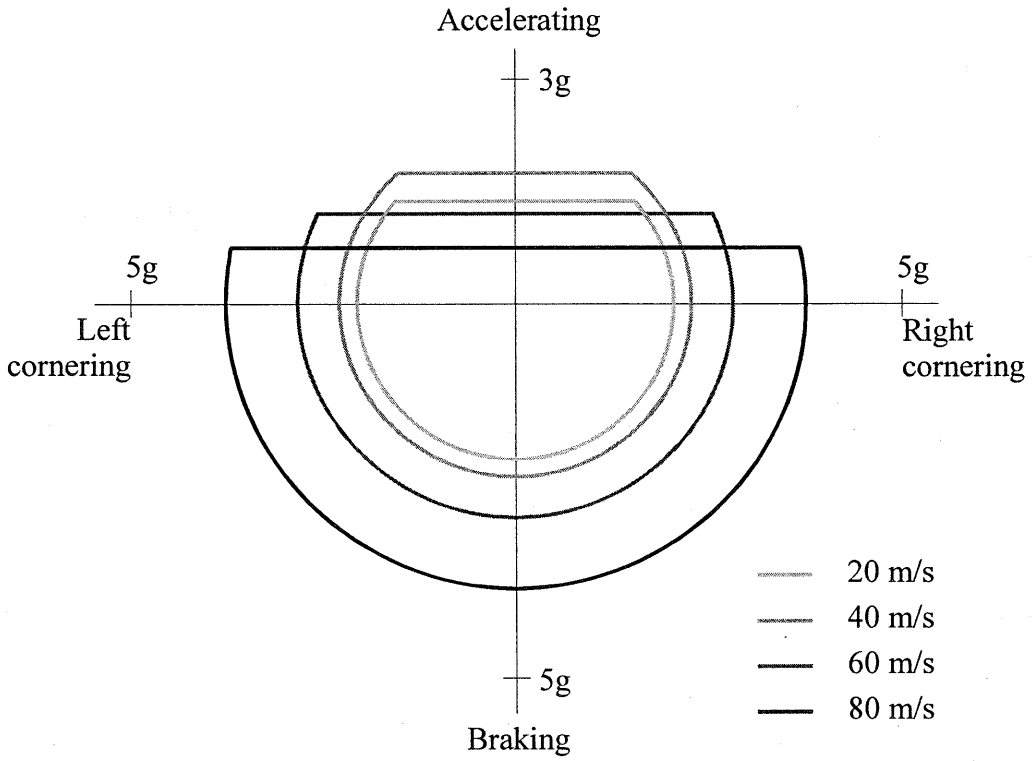


Figure 1.1 Qualitative representation of the g-g diagram of a race car with downforce.

Several computer programs have been developed using the concept of the performance envelope to estimate the race car lap time. The basic method assumes the racing line known by some means and divides the circuit into arbitrarily short segments with constant curvature where the vehicle operates at its steady-state performance limits. At the simplest level the vehicle may be treated as a mass point whose maximum performance is determined by the lateral and longitudinal tyre friction limits, and the maximum engine and braking power available. More complex vehicle models may be employed, providing a more accurate description of tyre forces, lateral and longitudinal load transfer, aerodynamic forces etc. The lap simulation starts by identifying the apex of each turn, usually the point where the curvature of the path has a local maximum. Here, the vehicle is assumed to be in pure steady-state cornering condition and the velocity may be easily evaluated. Then the simulation proceeds forward and backward from each of the apexes by finding the maximum positive or negative acceleration compatible with the curvature of the neighbouring segments and the local vehicle velocity. At the end of the simulation the various arcs will meet at the points where the vehicle longitudinal control switches from driving to braking. Examples of lap time

simulation programs based on this method may be found in (Griffiths, 1992, Gadola et al., 1996). A different approach to represent the vehicle performance boundaries is the Moment Method described in (Milliken and Milliken, 1995). This method is based on the study of the steady-state forces and moments which are available for linearly and angularly accelerating the vehicle at any working condition, e.g. combination of vehicle side slip angle, steer angle, longitudinal velocity and longitudinal acceleration. Their latest lap time simulation program uses this technology to work out the steady-state trimmed solution for each of the constant radius segments which make up the racing line.

However accurate the prediction of the lap time from any of these computer simulations may be, they are all limited by the very idea on which they are based. The steady-state approximation is often not acceptable when studying the response to vehicle design parameter changes in terms of total performance. In fact, many parameters have a significant effect on the vehicle transient behaviour. An example is the yaw moment of inertia, which has a significant influence when the vehicle is changing direction. None of the methods described above could be used to study this problem, as they only consider trimmed steady-state conditions. Furthermore, the assumption that the racing line is known, is seen as a constraint, when the task is to evaluate the maximum possible performance of a racing car. As is well known from practical experience, a race car has to be set up differently for different circuits. Besides, when the set-up is altered, the driver has to change his driving strategy in order to bring the car to its new limits. In other words the circuit geometry, the vehicle set-up and the optimal driving strategy are strongly coupled with each other.

In recent works, some authors have moved on from the steady-state approach and have set up lap time simulation techniques as path following tasks and implemented transient vehicle models, by either using driver models (Jennings, 1996) or inverse dynamic simulations (La Joie, 1994). But neither of these methods yields satisfactory results. Firstly, coupling a driver model with a vehicle model may change the dynamic properties of the system to such an extent that the solution is no longer representative of the real problem. Secondly, it is not clear how the maximum performance of the vehicle is simulated, which is fundamental when comparing different vehicle configurations. In the absence of an optimisation strategy, this would have to be done manually by specifying more challenging manoeuvres, which is not so convenient. A more advanced work from this point of view is from (Metz and Williams, 1989). They used Optimal Control theory to find the minimum manoeuvring time for a transient vehicle model along an imposed path. This, though, is not yet a breakthrough for a more accurate lap simulation, as specifying the vehicle path over-simplifies the problem. A new strategy is needed, which recognises the true nature of the minimum time manoeuvring problem, as well as including the transient phases.

1.2. THE NATURE OF THE MINIMUM TIME MANOEUVRING PROBLEM

"Driving a car as fast as possible (in a race) is all about maintaining the highest possible acceleration level in the appropriate direction."

(Peter G. Wright, quoted in Milliken and Milliken, 1995)

Figure 1.2 qualitatively describes the meaning of this quote by Peter Wright, former Technical Director of Team Lotus. Circuit racing driving may be described as a smooth combination of braking, driving and cornering. When arriving to a corner, a racing driver applies maximum braking torque when still proceeding on a straight line, but then he/she maintains the braking ideally until he/she reaches the apex of the turn, progressively easing off the brakes while increasing cornering. During this phase, the acceleration vector which is applied at the vehicle centre of gravity has initially only a longitudinal component in the vehicle reference axes system, and successively rotates towards the inside of the corner so that a variable combination of longitudinal and lateral accelerations acts on the car. Only at the apex, and for an instant, is the car ideally in pure steady-state cornering. Immediately after the apex, the driver applies driving torque and accelerates progressively out of the turn. Hence, the car is again subjected to a combination of lateral and longitudinal accelerations until it reaches the next straight.

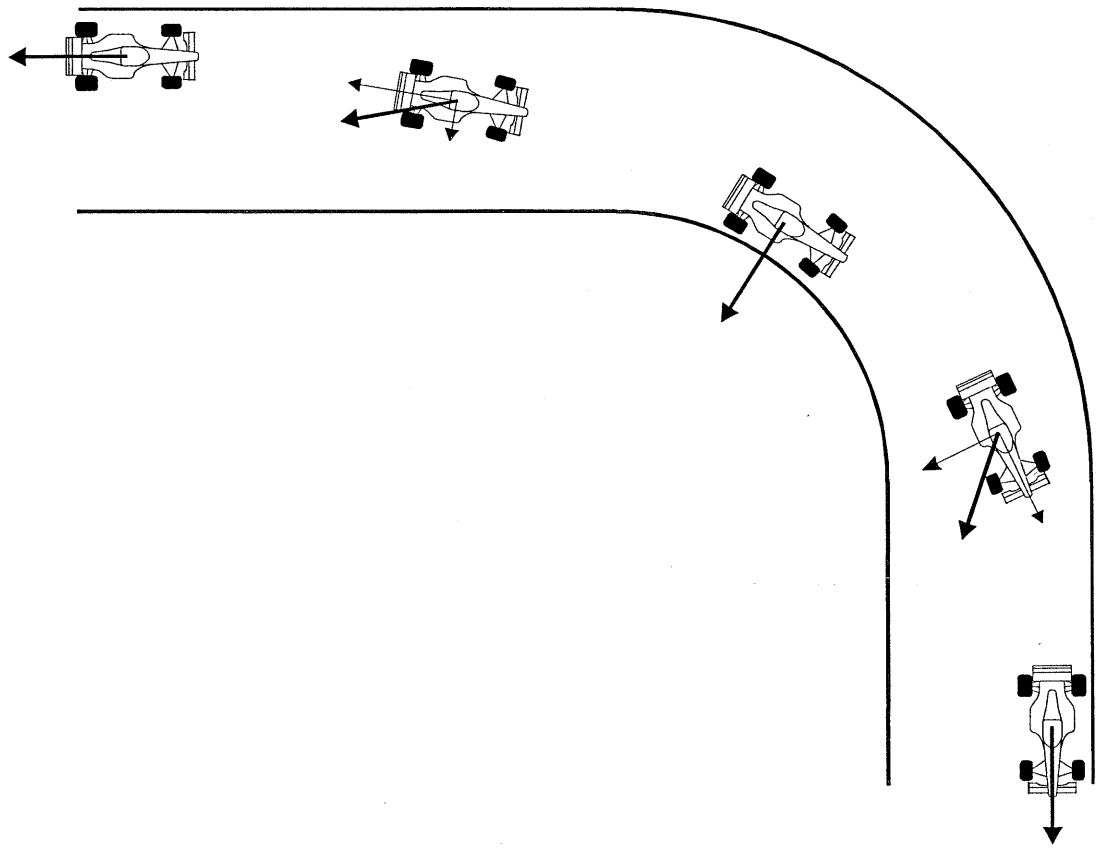


Figure 1.2 Acceleration vector of a race car mass centre through a corner.

At this stage we may say that if there exists an optimal history of the acceleration vector in terms of its magnitude and direction which yields the minimum manoeuvring time, it must be linked with a particular trajectory which has the right properties, i.e. its curvature, to make it possible. Furthermore, the maximum acceleration applicable to the vehicle is constrained by the saturation properties of the tyre forces in combined longitudinal and side slip conditions. Therefore, the optimal racing line, known in the racing world as "the groove", appears to be determined for any particular circuit geometry by the physical properties of the system, i.e. the vehicle. It is clear also that

any change in the vehicle configuration implies a different utilisation of the tyre forces and results in a different optimal racing line. Hence, the ideal software tool for simulating the maximum lap performance must also find the optimal driving strategy.

Not only does the saturation of the tyre forces limit the performance envelope of a race car, but in some conditions it is not possible for the vehicle to fully use the maximum tyre lateral forces available for cornering. Steady-state lap time simulation methods, which have been reviewed in the previous paragraph, assume that the tyre lateral forces are always used to the maximum of their availability to counteract the vehicle lateral acceleration. This, though, does not always happen in reality as is qualitatively shown in figure 1.3. As the vehicle approaches the first right-hand corner, the front tyres begin to generate lateral force with a time lead with respect to the rear tyres, sufficient to create enough moment about the yaw axis in order to angularly accelerate the vehicle (phase I). After this transient phase, the vehicle reaches the steady-state cornering condition where the yaw rate remains constant and the front and rear axle lateral forces are both directed towards the inside of the corner (phase II). When approaching the next left-hand corner, the vehicle yaw rate varies rapidly towards the opposite direction. Therefore, the front axle lateral force must again have a consistent time lead with respect to the rear axle lateral force in order to generate sufficient yaw moment. Hence, there will be a transient where the front and rear lateral forces are acting in opposition (phase III). Finally, the vehicle progresses towards the next apex where it will reach again steady-state cornering condition (phase IV). It is clear then that steady-state lap time simulation methods are bound to over-estimate the lap time by neglecting the transient phases where the car can not be on the edge of the performance envelope.

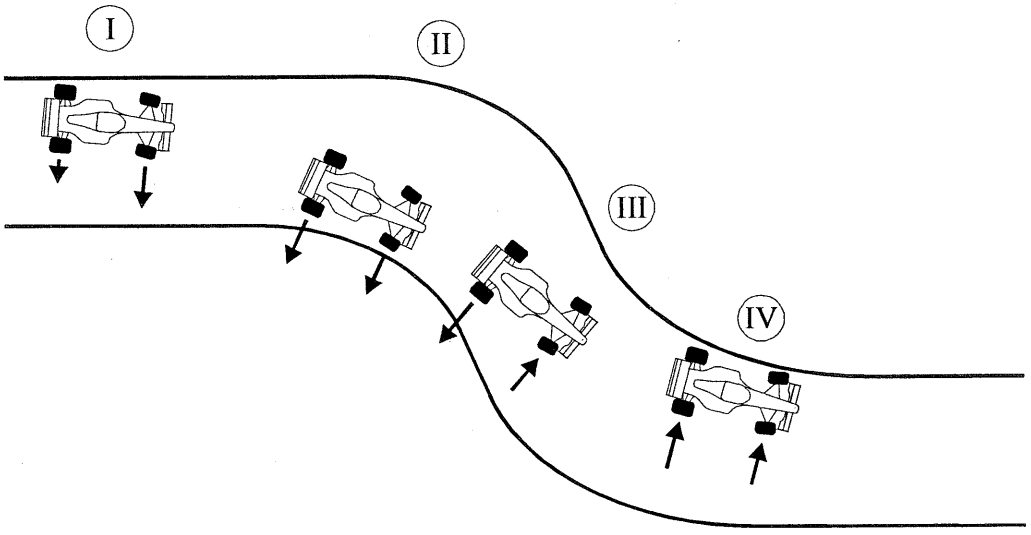


Figure 1.3 Axle forces through a lane change manoeuvre.

1.3. THE OPTIMAL CONTROL MINIMUM TIME MANOEUVRING PROBLEM

A significant advance in race car performance evaluation is the formulation of the minimum time manoeuvring problem as one of Optimal Control. The task may be

defined as follows. For a given transient vehicle dynamics model and a given track model, find the control histories, i.e. directional and longitudinal controls, which allow the vehicle to traverse a lap in as short a time as possible without violating the road boundaries. An early work in this field is due to (Fujiuka and Kimura, 1992), who employed a simple transient vehicle dynamics model to represent cars with a variety of driving and steering configurations. They applied the Sequential Conjugate Gradient Restoration Algorithm (Wu and Miele, 1978a and 1978b) to minimise the time for a hairpin manoeuvre with start and finish conditions chosen in such a way that the car stayed on the track without needing constraints. However, this would not happen so conveniently in general. Quite recently (Hendrikx et al., 1996) made an important contribution to this subject. Their work can be considered the first significant formulation of the minimum time optimal control problem of a general vehicle manoeuvre. They employed a more advanced vehicle model featuring 3 degrees of freedom, an engine torque curve, non-linear tyre model and steady-state load transfer and aerodynamic forces. The condition for the car to stay on the road was introduced as state constraints and the track was described through the co-ordinates of its centre line and its width. They solved the problem for simple vehicle manoeuvres, i.e. a single lane change, by applying Optimal Control Theory and Pontryagin's Minimum Principle (Kirk, 1970, Bryson and Ho, 1975). This approach involves the setting up of the adjoint equations (which represent the sensitivity of the objective, i.e. the manoeuvre time, with respect to the system states) and results in a non-linear, two point boundary value problem. Their results showed how this method is capable to evaluate the vehicle maximum performance in transient conditions and that the optimal driving strategy is different for different car configurations. The main weakness of the work is in the implementation of the Gradient Descent algorithm, as is described in (Kirk, 1970), for the solution of the Optimal Control problem. This algorithm is rather robust with respect to inaccurate starting values, but it is the least efficient in terms of speed of convergence. Very recently, (Cossalter et al., 1999) applied a similar approach for the evaluation of motorcycle manoeuvrability. They employed, though, a more efficient algorithm for solving the two point boundary value problem based on a relaxation method as described in Numerical Recipes (Press et al., 1992). Finally, (Allen, 1997) developed a solution equivalent to that of Hendrikx using a Direct Shooting method for solving the minimum time Optimal Control problem, using the algorithm described in (Kraft, 1994). This method showed many advantages compared to the traditional application of Pontryagin's Minimum Principle and offered the possibility to be computationally more efficient than Hendrikx's approach.

This research is aimed to develop the Optimal Control solution of the minimum time manoeuvring problem, to extend its application into a full lap time simulation and to apply it to vehicle design studies.

1.4. DRIVER MODELLING

Lap time optimisation begins from a set of initial controls which allows the car to complete a lap without excessively violating the constraints. We may anticipate that the more accurate the initial controls are, the easier will be the work of the optimisation algorithm. The problem of finding feasible initial controls is set up as a path following task and a part of this research is dedicated to the development of a driver model.

Several such models have been described in the literature and a short review of the main ideas used follows.

The earliest driver model of note was the "cross-over" model (Weir and McRuer, 1968). The cross-over model depends on representing driving as a small perturbation, regulation task. As is based on experimental testing of man and machine interaction, it nicely represents the adaptation of a person to the machine response in a regulation task. However, only in rare situations does driving resemble such conditions, the main one of which is travelling in a straight line under cross-wind disturbance. Since the main concern here is with path following, the cross-over model will not be discussed further.

A different approach to driver modelling involves the use of preview information of the road ahead. Driver models based on this concept were reviewed by (Guo and Guan 1993). As indicated there, substantial progress in driver modelling in a linear world was made by (MacAdam, 1981). He represented the driver as an optimal preview controller, constructing a path error functional by previewing the road over a known preview distance, and minimising a weighted integral of squares of differences between the previewed path points and the corresponding estimated lateral positions of the vehicle over the preview distance. The main weakness of the theory, as pointed out by Guo and Guan, is the need to presume a constant control input over the preview interval in order to estimate the vehicle position. Notwithstanding this presumption, the theory, in single preview point form, was shown to more or less subsume the cross-over model for a regulation task and to be quite capable in respect of path following control. Many detailed developments from this starting point are described by the reviewers, concentrating on the relatively elementary single preview point cases. This particular feature is considered unrealistic and unsatisfactory, since it is clear that if the preview point is a long way in front of the vehicle, it will be inappropriate to act on the preview information at the time of its acquisition and the information has gone by the time it is useful. On the other hand, if the preview point is very close to the vehicle, control will necessarily be very poor. It is always the case that with single point preview models, the first concern in choosing parameter values is to set a preview distance which is reasonable. One cannot imagine a real driver using single-point preview.

In parallel with work on driver steering models, optimal linear preview control theory (Kwakernaak and Sivan, 1972) has been applied extensively to active vehicle suspensions (Sharp, 1995, Sharp and Prokop 1995). Both continuous time and discrete time approaches have been followed, the latter leading to rather simpler interpretations. It has been established that optimal linear preview control involves diminishing returns, i.e. for any task, there is an amount of preview beyond which further preview is of no value to the controller. If such further preview were available, the controller would take no notice of it. In discrete time, optimal use of the preview data involves forming a weighted sum of the preview error samples and using the sum for control. In the complete optimal scheme, this sum is combined with a corresponding sum of weighted state feedback terms to give the required control. If the preview is foreshortened, the optimal gains are unaltered. What happens to the optimal control is that it is just the remainder from the infinite preview case, with the lost preview information simply missing. Finally, the gain sequence associated with the preview error samples reflects the dynamics of the controlled plant, indicating the use of the preview control to motivate the system by operating preferentially on its eigenmodes. Thus, if the plant has a lightly damped mode, and the control bandwidth is sufficient to excite that mode, the optimal preview control gain sequence will show the oscillatory pattern of the

eigenmode. On the other hand, if the plant is very well damped, the gain sequence will appear more like an exponential convergence on zero gain for large preview distance.

If this set of ideas from suspension control is taken across into the world of driver modelling, the following picture emerges. While the vehicle is operating in its linear region, the linear discrete time optimal preview control theory applies directly. It indicates that the driver will acquire path preview information by looking ahead of the vehicle, forming lateral position error estimates by comparison of the path ahead with points along an imaginary optical lever projecting forwards from the vehicle, as shown in figure 1.4 (a). The driver will also have state error information from the relation between the present position and that desired (perfectly on the path). These error estimates will be weighted and summed, the preview weighting values reflecting to some extent the vehicle lateral dynamics. The state feedback gains are those which belong to the non-preview case. The control scheme is illustrated in figure 1.4 (b).

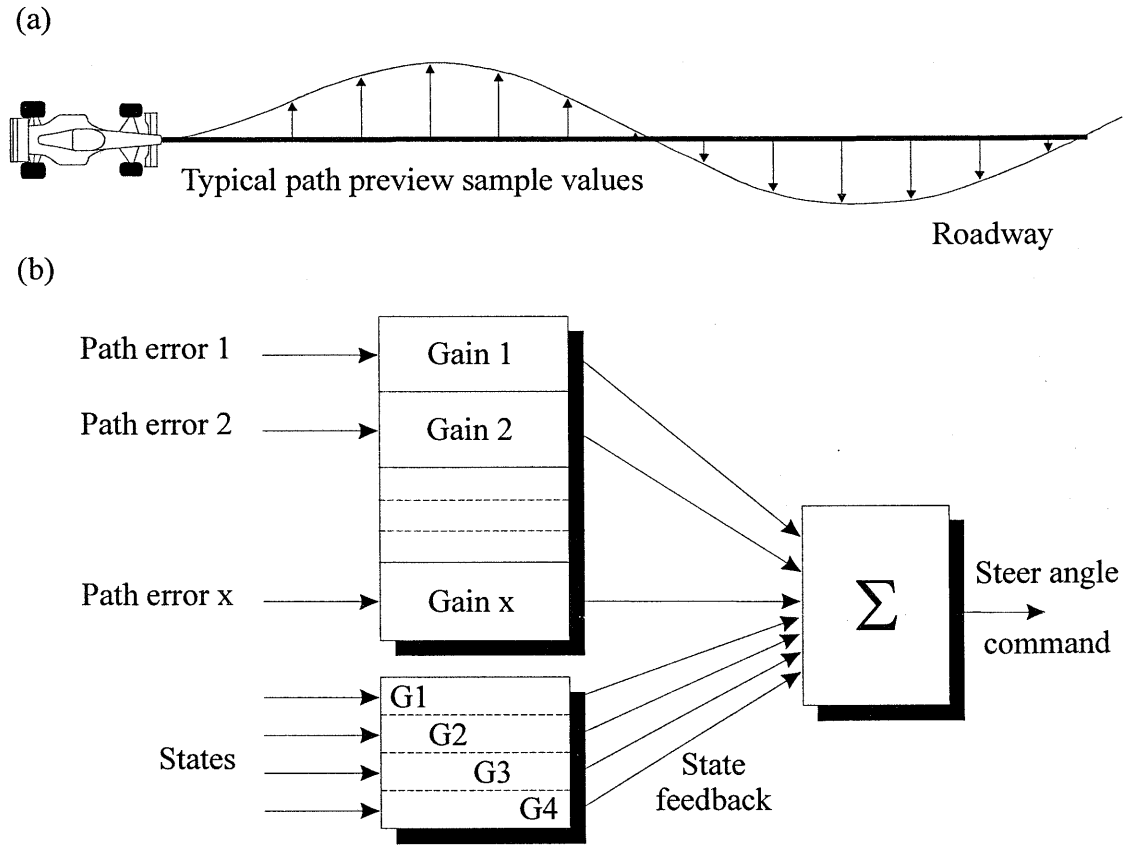


Figure 1.4 Notional structure of optimal linear preview driving controller.

In vigorous manoeuvring, tyre forces saturate and imply non-linear vehicle behaviour. To deal with this case, it can be observed that the structure of the control scheme in figure 1.4 is that of a linear neural network. It is apparent that the saturating non-linearity of the real vehicle can be accommodated by the replacement of the linear network by a non-linear one. It can be anticipated that the optimal linear gains will provide a starting point for the calculation of good network parameters, refinement coming from a learning process.

In more recent work, (MacAdam and Johnson, 1996) have shown interest in neural network structures for driving control. They argue for more than one preview

point, using three in fact, and they attach value to the possibility of combining preview sample values by differencing (say) to obtain implied path orientation data. They also value time-delayed versions of the path preview samples and feed some time-delayed observations into their control network alongside the three current preview sample values. The association made above between linear optimal discrete time preview suspension control and driving suggests a different interpretation of the value of this time-delayed data, since the optimal suspension controller does not calculate differences between preview samples or combine them in other ways. Neither does it do anything equivalent to using the path preview samples to compute the future path of the vehicle, based on some model of the vehicle dynamics. The structure of the optimal linear controller suggests that it is not necessary to perform these operations on the preview data, provided that the preview points are sufficiently closely spaced.

In this research the above framework is applied to set up a particular driver model, which deals with the non-linear vehicle operating regime. Saturation properties are introduced into the controller, to match the saturation properties of the vehicle tyres to a certain extent. This implies that the methods of optimal linear preview control may not be applied in order to find the model parameters. Therefore, in its current form, the model parameters are derived in an ad hoc fashion, based on intuition and not on any formal optimisation scheme.

1.5. CIRCUIT MODELLING

The model of a circuit is essentially described by the following (non-independent) parameters (figure 1.5):

- The co-ordinates of its centre line in a reference axes system fixed in space, x_t, y_t;
- The curvature of the centre line, k_t, or its local radius r_t;
- The tangent angle of the centre line, ψ_t;
- The road width, w_t.

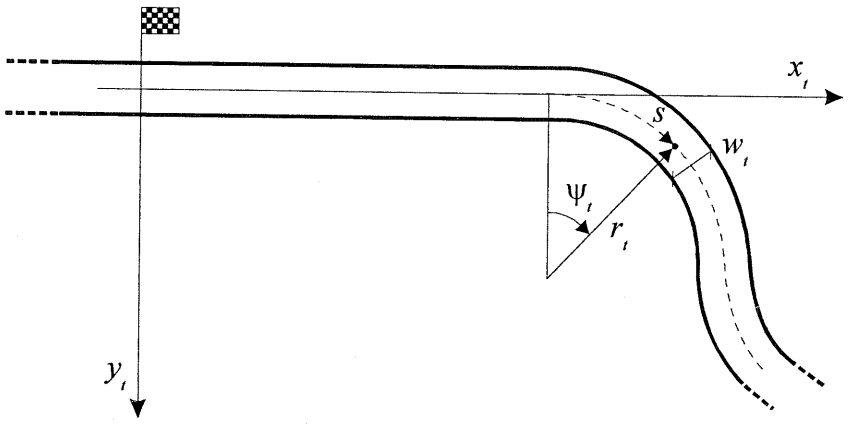


Figure 1.5 Circuit model description.

These data are expressed as functions of the independent path co-ordinate s, i.e. the distance travelled along the road centre line from the start-finish line. Usually, the track

data are supplied to the vehicle model as look-up tables. Finally, complementary information may be supplied, e.g. road elevation, road camber angle, variations in the friction coefficient of the road surface, etc.

It is unlikely to obtain the necessary track data in the proper digital format from the various circuits. More common is to obtain a hardcopy of a detailed map of the track, as was the case for the Suzuka International Race Course (figure 1.6). Here, it was possible to derive the description of the circuit as a sequence of straight lines and constant radius arcs, which led to its mathematical description straightforwardly. Other methods for deriving a digital map of a circuit involve scanning the picture of the track and saving it into a bit map format. Then, using image processing techniques, the co-ordinates of the road course within the picture may be detected and the digital map so obtained only needs to be re-scaled to the full size of the circuit.

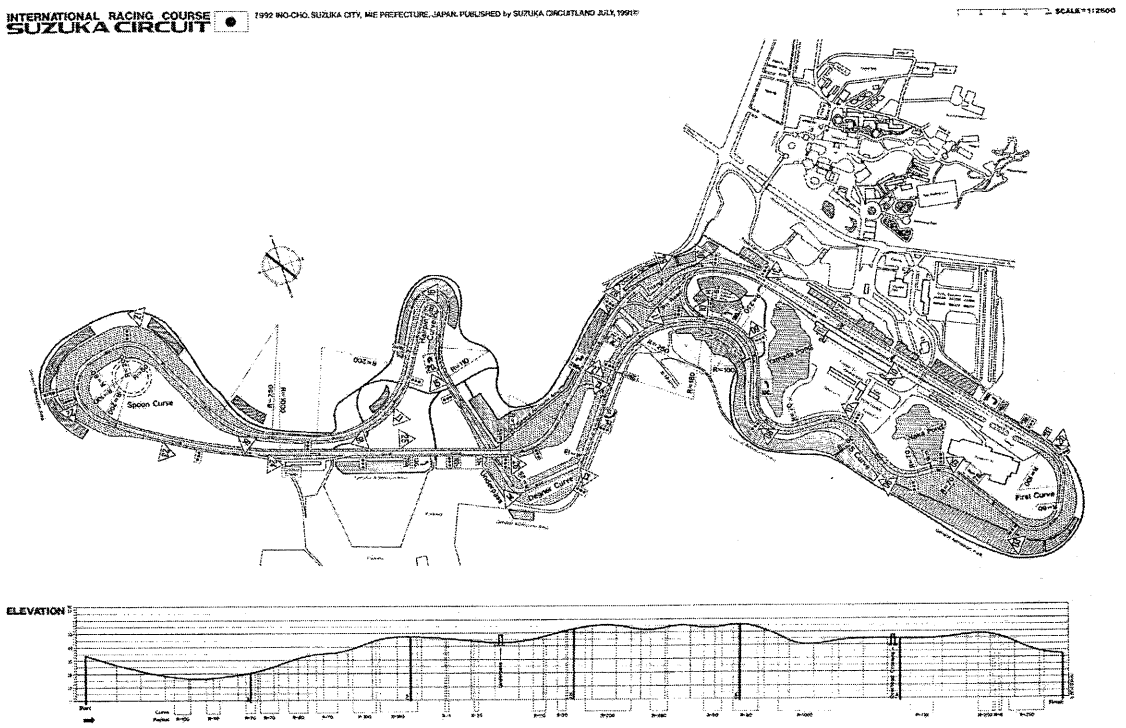


Figure 1.6 The map of Suzuka International Race Course.

A different strategy for circuit modelling involves using the on board measured data to reconstruct the vehicle path. Longitudinal velocity and lateral acceleration are used to obtain an estimate of the curvature of the racing line as a function of the travelled distance. Starting from curvature information, the path tangent and the path co-ordinates may easily be evaluated. However, the main problem here lies in the noise which accompanies the experimental data. The error in the measurements is enormously amplified by the double integration which is involved in the racing line reconstruction procedure. Special strategies are needed in order to obtain a closed trajectory which sufficiently approximates the real racing line. This problem will be discussed in detail in section three. A final consideration has to be done regarding the use of the approximated racing line as a circuit model, though. Simply taking the racing line as the road centre line for the circuit model obviously leads to a much faster circuit with a

different geometry from the real one. The ideal strategy to quickly obtain a good approximation of the real circuit layout could then be to have the driver completing a few laps at lower than race speed maintaining the car as close as possible to the centre of the road. Alternatively, Global Positioning System technology may be employed for this purpose.

1.6. THESIS OUTLINE

Chapter One: Introduction

Chapter Two: Optimal Control Theory
The general theory of Optimal Control relevant to this research is outlined in this section, and the various solution methods are reviewed.

Chapter Three: Racing Line Reconstruction from Telemetry Data
An algorithm for racing line reconstruction using on board measured data is described.

Chapter Four: The Driver Model
A new mathematical model for vehicle steer control, derived from Linear Optimal Control theory, is described. Various methods for longitudinal vehicle control are also presented, making for a complete set of path following strategies.

Chapter Five: Path Following Results
A set of results related to the two previous chapters is presented. Various path following tasks are shown, both regarding moderate g manoeuvres as well as lap simulation at racing speed, in order to prove the effectiveness of the driver model.

Chapter Six: Lap Time Optimisation
Definition of the lap time optimisation problem as a minimum time Optimal Control problem. Choice of the suitable solution method.

Chapter Seven: Evaluating Derivatives
Starting from the problem of obtaining fast and reliable derivatives needed to solve the optimisation problem, this chapter goes on focusing on the theory of Automatic Differentiation and its advantages over the traditional numerical differentiation techniques.

Chapter Eight: Introducing FastLap
The software tool for lap time optimisation developed in the course of this research is described in this section.

Chapter Nine: Optimisation Results
The results from the lap time optimisation program are described and discussed in this section. Various parameter sensitivity study cases are presented.

Chapter Ten: Conclusions and Further Work

Chapter 2.

Optimal Control Theory

The objective of Optimal Control Theory is to determine the control input signal that will cause a plant to minimise (or maximise) some performance criterion while satisfying a set of physical constraints specific for the plant. Optimal Control Theory has its foundation in the Calculus of Variations. Queen Dido of Carthage was apparently the first person in history to attack a problem which can be solved by applying variational calculus. Dido, having been promised all the land she could enclose with a bull's hide, cleverly cut the hide into many lengths and tied the ends together. The problem was then to find the closed curve with fixed perimeter that enclosed the maximum area. Calculus of Variation enables us to readily find that she should have chosen a circle.

This section presents a review of the Theory of Optimal Control which is relevant for the development of this research. Firstly, the general form of an Optimal Control problem for non linear systems is introduced, together with some useful definitions. Then, the Pontryagin's Minimum Principle, which states the first order, necessary conditions for optimality, is derived (Kirk, 1970, Bryson and Ho, 1975). This involves the application of some of the principles of Calculus of Variations. A summary of such principles may be found in (Kirk, 1970, § 4.1). A family of methods for solving Optimal Control problems, known as *indirect methods*, rely on the application of the Pontryagin's Minimum Principle and involve the setting up of the equations representing the optimality conditions, which constitute a non-linear, two point boundary value problem. A brief description of these methods is also included.

A different class of methods aims to solve an Optimal Control problem directly by converting the original continuous problem into a discrete approximation and applying non-linear programming techniques. Such methods, known as *direct methods*, rely on the application of the theory for the constrained minimisation of non-linear, multi-variable functions, the Kuhn-Tucker conditions (Fletcher, 1987). The relevant theory of non-linear programming is presented in the second part of this chapter and the various direct methods of optimisation are reviewed. Finally, the connection with the classical theory of optimisation is established by showing that the Kuhn-Tucker conditions applied to a discretised Optimal Control problem are, in fact, a discrete version of the Pontryagin's Minimum Principle (Enright and Conway, 1990).

2.1. Optimal Control problem formulation

The formulation of a general Optimal Control problem requires:

- A mathematical model of the plant to be controlled;
- A statement of the physical constraints;
- A performance criterion.

The definition of the problem involves the use of *functionals*. A functional J is a rule of correspondence which assigns to each function $x(t)$, member of a certain class Ω, a unique real number. Ω is called the *domain* of the functional, and the set of real numbers associated with the functions in Ω is called the *range* of the functional. Intuitively, a functional may be seen as a "function of functions". An example of a functional is the integral of a function over a finite interval:

$$J(x(t)) = \int_{t_0}^{t_f} x(t)\,dt \qquad \text{Eq. 2.1}$$

2.1.1. The mathematical model

The objective of the mathematical model is to predict the response of the plant to all anticipated control inputs. In this review of Optimal Control Theory we shall restrict the discussion to non-linear, time-variant systems described by first-order, ordinary differential equations. A mathematical model for a generic system of this kind, representing the rate of change of its states with respect to the time, may be stated as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \qquad t \in [t_0, t_f] \qquad \text{Eq. 2.2}$$

Here, $\mathbf{x}$ is the set of system state variables, or simply system states, and $\mathbf{u}$ is a vector function which returns the control signal applied to the plant. We shall refer to any time history of state and control variables defined within the interval $[t_0, t_f]$ as *state trajectory* and *control history* respectively.

2.1.2. The physical constraints

The physical constraints in an Optimal Control problem are intended to limit the range of the state and control variables within values which are meaningful for the plant and for the problem which is being analysed. To illustrate some typical constraints, we may refer to the minimum lap time optimisation problem. In this case, the longitudinal control for the race car is limited by the maximum driving and braking torque available. The lateral control is also limited by the geometry of the steering system to a maximum steer angle. Finally, the co-ordinates of the vehicle centre of mass must lie between the road boundaries for the entire lap.

We shall refer to a general physical constraint involving the state trajectory and the control history with the following statement:

$$c(t) = c(\mathbf{x}(t), \mathbf{u}(t), t) \le 0 \qquad t \in [t_0, t_f] \qquad \text{Eq. 2.3}$$

Alternatively, we may have local constraints defined as functions of the state and control variables at a particular time t_i:

$$c_i = c_i(\mathbf{x}(t_i), \mathbf{u}(t_i), t_i) \leq 0 \qquad t_0 \leq t_i \leq t_f \qquad \text{Eq. 2.4}$$

In the forthcoming theoretical developments, the entire set of control and state constraints will be indicated with **c**. Finally, a different kind of constraint is represented by constant control bounds. The definition reads:

$$\mathbf{u}_L \leq \mathbf{u}(t) \leq \mathbf{u}_U \qquad t \in [t_0, t_f] \qquad \text{Eq. 2.5}$$

A control history which satisfies the control constraints during the entire time interval $[t_0, t_f]$ is called an *admissible control*. We shall define the set of admissible controls by U, and the notation $\mathbf{u} \in U$ means that **u** is admissible. In the same way, a state trajectory which satisfies the state variable constraints during the entire time interval $[t_0, t_f]$ is referred to as a *feasible trajectory*. The set of feasible trajectories is denoted by X, and the notation $\mathbf{x} \in X$ means that **x** is feasible.

2.1.3. The performance measure

The objective of an Optimal Control problem is to minimise (or maximise) a quantitative measure of the performance of the plant. The most general definition for the performance measure involves a function of the final system states as well as a functional of the state trajectories and the control histories:

$$J = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)\, dt \qquad \text{Eq. 2.6}$$

An alternative definition of the performance measure, that we shall use later on, is the following. Assuming that the system has p states, we may define an additional state variable x_{p+1} and associate the second term in the right hand member of Eq. 2.6 to it:

$$x_{p+1}(t) = \int_{t_0}^{t} g(\mathbf{x}(t), \mathbf{u}(t), t)\, dt \qquad \text{Eq. 2.7}$$

The integral part of Eq. 2.6 may then be substituted with the final value of the additional state variable and the performance measure simply reads:

$$J = \bar{h}(\bar{\mathbf{x}}(t_f), t_f) \qquad \text{Eq. 2.8}$$

where $\bar{\mathbf{x}}$ is the augmented set of system states:

$$\bar{\mathbf{x}} = \{x_1 \quad x_2 \quad \ldots \quad x_p, x_{p+1}\} \qquad \text{Eq. 2.9}$$

The performance measure characterises the different types of Optimal Control problem, e.g. minimum time problems, minimum control effort problems, path tracking problems, etc. For the case of the minimum lap time optimisation problem the performance measure is simply the time that it takes to traverse a lap of the circuit and its expression reads:

$$J = \int_{t_0}^{t_f} dt \qquad \text{Eq. 2.10}$$

2.1.4. The Optimal Control problem

The task in an Optimal Control problem is to find an admissible control **u*** which causes the system described by Eq. 2.2 to follow a feasible trajectory **x*** which minimises the performance measure J:

$$\min_{\mathbf{u}} \quad J = h\left(\mathbf{x}(t_f), t_f\right) + \int_{t_0}^{t_f} g\left(\mathbf{x}(t), \mathbf{u}(t), t\right) dt$$

$$\begin{aligned} \textit{subject to}: \quad & \dot{\mathbf{x}}(t) = \mathbf{a}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \\ & \mathbf{c} \le 0 \\ & \mathbf{u}_L \le \mathbf{u}(t) \le \mathbf{u}_U \end{aligned} \qquad \text{Eq. 2.11}$$

$$\textit{for all } t \in [t_0, t_f]$$

The control **u*** is the *optimal control* and the corresponding trajectory **x*** is the *optimal trajectory*.

2.2. THE NECESSARY CONDITIONS FOR OPTIMALITY

In this paragraph we shall derive the first order, necessary conditions for optimality, i.e. the conditions that the state trajectory and the control history must satisfy when the performance measure J is on a relative *extremum*. The procedure is in all respects similar to the equivalent problem in calculus of finding an extremum of a function. Consider a continuous and differentiable function of a single variable $f(q)$. The theory of calculus states that the necessary condition for q^* to be an extremum is that the first derivative of $f(q)$ vanishes when $q \to q^*$. If we write the increment of $f(q)$ for an arbitrarily small Δq, such increment may be approximated with the differential of $f(q)$:

$$f(q+\Delta q) - f(q) = df(q, \Delta q) + o(\Delta q) \cong df(q, \Delta q) = \frac{df(q)}{dq} \cdot \Delta q \qquad \text{Eq. 2.12}$$

where $o(\Delta q)$ represents the higher order terms in the series expansion of $f(q)$ when Δq

tends to 0. Hence, the necessary condition for $f(q)$ to have an extremum at $q = q^*$ is also that its differential vanishes:

$$df(q^*, \Delta q) = 0 \qquad \text{Eq. 2.13}$$

In the corresponding problem of Calculus of Variations the task is to define the first order approximation to the increment of a functional. Then the necessary condition for having an extremum will be that such approximation be equal to zero. For a differentiable functional $J(\mathbf{x})$ we may write its increment as follows:

$$J(\mathbf{x}+\delta\mathbf{x}) - J(\mathbf{x}) = \Delta J(\mathbf{x}^*, \delta\mathbf{x}) = \delta J(\mathbf{x}, \delta\mathbf{x}) + o(\delta\mathbf{x}) \qquad \text{Eq. 2.14}$$

Here, $\delta J(\mathbf{x}, \delta\mathbf{x})$ is called the *variation* of a functional and is the equivalent of the differential of a function in the theory of calculus. The function $\delta\mathbf{x}$ is an arbitrarily small perturbation distributed along the trajectory $\mathbf{x}$. Figure 2.1 visualises qualitatively a generic perturbation δx for the case of a scalar function.

When $\delta\mathbf{x}$ vanishes, the increment of a functional may be approximated with its variation. Then, the necessary condition for J to have an extremum in $\mathbf{x}^*$ is that its variation must vanish on $\mathbf{x}^*$, that is:

$$\Delta J(\mathbf{x}^*, \delta\mathbf{x}) \cong \delta J(\mathbf{x}^*, \delta\mathbf{x}) = 0 \qquad \text{Eq. 2.15}$$

for all admissible $\delta\mathbf{x}$ (which means that if Ω is the domain of J, $\mathbf{x}^* + \delta\mathbf{x}$ must still be a member of such domain).

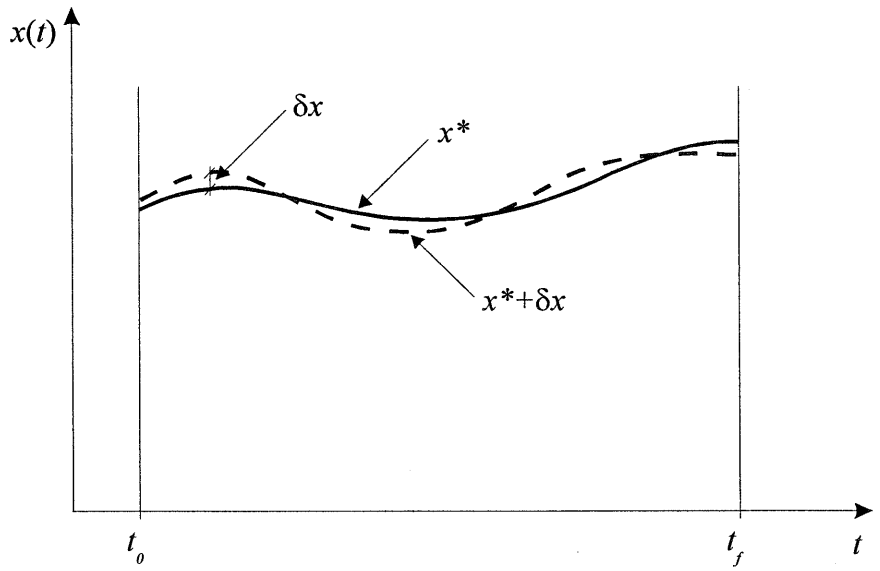


Figure 2.1 A function with a neighbouring curve.

At this stage it is important to point out that even if we were able to find a curve $\mathbf{x}^*$ which satisfies Eq. 2.15, there is no certainty that such curve would be an extremum, as Eq. 2.15 only states a necessary condition. Furthermore, even if $\mathbf{x}^*$ were an

extremum, nothing could be said on whether it is a local minimum or a local maximum. Finally, it is not even guaranteed that the functional is differentiable at the extremum **x***. The analogous case in calculus is the function:

$$f(q)=|q| \qquad q\in[-\infty,+\infty] \qquad \text{Eq. 2.16}$$

This function obviously has a global minimum when $q = 0$, but it is not differentiable at this point.

As well as in the case of the theory of calculus, where we would look at the second order derivative of a function in order to establish necessary and sufficient conditions for either a local minimum or maximum to occur, the second order variation of a functional may be defined, and necessary and sufficient conditions of optimality may be derived (Bryson and Ho, 1975). However, this involves a rather complex manipulation of the problem equations which does not lead to something practical. Conversely, the necessary conditions for optimality provide a convenient starting point to use for searching a solution. We shall then use our knowledge of the physical properties of the plant to judge whether or not the solution that we eventually find is optimal.

As the scope of this section is to review the theory of Optimal Control which is relevant to this research, we shall not derive the necessary conditions for optimality for all the possible optimisation problems. Firstly, we shall consider problems without state and control constraints, and with fixed end time. The latter condition may seem to be a contradiction for the minimum time problem, as t_f in Eq. 2.10 is obviously not a constant. However, we may anticipate that the minimum lap time optimisation problem may be posed in a simpler way by substituting the time t with the travelled distance s as independent variable. This allows to evaluate the performance measure over a fixed interval, e.g. the lap distance, and justifies the above assumption. Then, we shall introduce the control boundaries and we shall derive the Pontryagin's Minimum Principle as a general statement of the necessary conditions for optimality.

2.2.1. Unconstrained Optimal Control problems with fixed end time

Let us consider a system with p state variables and q control variables described by the following set of first-order, non-linear differential equations:

$$\dot{\mathbf{x}}(t)=\mathbf{a}(\mathbf{x}(t),\mathbf{u}(t),t) \qquad t\in[t_0,t_f] \qquad \text{Eq. 2.17}$$

We shall assume that the final time t_f is fixed and that the initial conditions are given and are fixed as well:

$$\mathbf{x}(t_0)=\mathbf{x}_0 \qquad \text{Eq. 2.18}$$

The task is to find a control history **u*** which causes the plant to follow a trajectory **x*** which minimises the performance measure:

$$J(\mathbf{u}) = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)\,dt \qquad \text{Eq. 2.19}$$

The functional J is assumed to be dependent only on the control $\mathbf{u}$. This is because any control history $\mathbf{u}$ univocally determines a state trajectory $\mathbf{x}$, and also because the initial states $\mathbf{x}_0$ as well as the final time t_f are fixed.

Let us firstly adjoin the differential equations to the performance measure by introducing p Lagrange multipliers $\boldsymbol{\lambda}(t)$ (for the sake of simplifying the expressions, from now on we shall omit the dependency of the control history, the state trajectory and the Lagrange multipliers on the time t):

$$\bar{J}(\mathbf{u}) = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} \left[g(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T \cdot (\mathbf{a}(\mathbf{x}, \mathbf{u}, t) - \dot{\mathbf{x}}) \right] dt \qquad \text{Eq. 2.20}$$

The last term in the integrand of Eq. 2.20 may be solved by parts in order to eliminate the state derivatives, and the result reads:

$$\int_{t_0}^{t_f} -\boldsymbol{\lambda}^T \cdot \dot{\mathbf{x}}\,dt = \boldsymbol{\lambda}^T(t_0) \cdot \mathbf{x}(t_0) - \boldsymbol{\lambda}^T(t_f) \cdot \mathbf{x}(t_f) + \int_{t_0}^{t_f} \dot{\boldsymbol{\lambda}}^T \cdot \mathbf{x}\,dt \qquad \text{Eq. 2.21}$$

Substituting Eq. 2.21 in Eq. 2.20 the following result is obtained:

$$\bar{J}(\mathbf{u}) = h(\mathbf{x}(t_f), t_f) + \boldsymbol{\lambda}^T(t_0) \cdot \mathbf{x}(t_0) - \boldsymbol{\lambda}^T(t_f) \cdot \mathbf{x}(t_f) + \int_{t_0}^{t_f} \left[g(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T \cdot \mathbf{a}(\mathbf{x}, \mathbf{u}, t) + \dot{\boldsymbol{\lambda}}^T \cdot \mathbf{x} \right] dt \qquad \text{Eq. 2.22}$$

Let us now introduce the *Hamiltonian* function, which is defined as:

$$H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) = g(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T \cdot \mathbf{a}(\mathbf{x}, \mathbf{u}, t) \qquad \text{Eq. 2.23}$$

By using the Hamiltonian function in Eq. 2.22 we obtain:

$$\bar{J}(\mathbf{u}) = h(\mathbf{x}(t_f), t_f) + \boldsymbol{\lambda}^T(t_0) \cdot \mathbf{x}(t_0) - \boldsymbol{\lambda}^T(t_f) \cdot \mathbf{x}(t_f) + \int_{t_0}^{t_f} \left[H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) + \dot{\boldsymbol{\lambda}}^T \cdot \mathbf{x} \right] dt \qquad \text{Eq. 2.24}$$

We may now proceed in writing the variation of the functional $\bar{J}(\mathbf{u})$ by differentiating Eq. 2.24 with respect to $\mathbf{u}$ and $\mathbf{x}$:

$$\delta \bar{J}(\mathbf{u},\delta\mathbf{u}) = \left[\left(\frac{\partial h}{\partial \mathbf{x}} - \boldsymbol{\lambda}^T\right)\cdot\delta\mathbf{x}\right]_{t=t_f} + \left(\boldsymbol{\lambda}^T\cdot\delta\mathbf{x}\right)_{t=t_0} + \int_{t_0}^{t_f}\left[\left(\frac{\partial H}{\partial \mathbf{x}} + \dot{\boldsymbol{\lambda}}^T\right)\cdot\delta\mathbf{x} + \left(\frac{\partial H}{\partial \mathbf{u}}\right)\cdot\delta\mathbf{u}\right]dt \qquad \text{Eq. 2.25}$$

Since the initial state values are fixed, the second term in the right hand member of Eq. 2.25 is equal to zero. Then, as the control history univocally determines the state trajectory, we may assume that the variation of the state trajectory $\delta\mathbf{x}$ depends on the variation of the control $\delta\mathbf{u}$. However, rather than trying to express $\delta\mathbf{x}$ as a function of $\delta\mathbf{u}$, we may choose the Lagrange multipliers in such a way that the terms in Eq. 2.25 which multiply $\delta\mathbf{x}$ vanish. In doing so, we obtain:

$$\dot{\boldsymbol{\lambda}}^T = -\frac{\partial H}{\partial \mathbf{x}} = -\frac{\partial g(\mathbf{x},\mathbf{u},t)}{\partial \mathbf{x}} - \boldsymbol{\lambda}^T\cdot\frac{\partial \mathbf{a}(\mathbf{x},\mathbf{u},t)}{\partial \mathbf{x}} \qquad \text{Eq. 2.26}$$

$$\boldsymbol{\lambda}^T = \frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}(t_f)) \qquad \text{Eq. 2.27}$$

For an extremum to occur, the variation of the functional must be zero for any arbitrary $\delta\mathbf{u}$. Therefore, after deleting all the terms equal to 0 in Eq. 2.25, this condition reads:

$$\delta \bar{J}(\mathbf{u},\delta\mathbf{u}) = \int_{t_0}^{t_f}\left[\left(\frac{\partial H}{\partial \mathbf{u}}\right)\cdot\delta\mathbf{u}\right]dt = 0 \qquad \text{Eq. 2.28}$$

However, Eq. 2.28 is satisfied only if:

$$\frac{\partial H}{\partial \mathbf{u}} = \frac{\partial g(\mathbf{x},\mathbf{u},t)}{\partial \mathbf{u}} + \boldsymbol{\lambda}^T\cdot\frac{\partial \mathbf{a}(\mathbf{x},\mathbf{u},t)}{\partial \mathbf{u}} = \mathbf{0} \qquad \text{Eq. 2.29}$$

Eqs. 2.26, 2.27 and 2.29 are known as the *Euler-Lagrange equations* in the Calculus of Variations. Eq. 2.26 are also known as the *co-state equations* and they represent the sensitivity of the performance measure for a variation in the state trajectory **x**. In summary, to find a control history $\mathbf{u}(t)$ which produces a stationary point of the performance measure J, we must solve the following $2p$ differential equations:

$$\dot{\mathbf{x}}(t) = \mathbf{a}(\mathbf{x},\mathbf{u},t) \qquad \text{Eq. 2.30}$$

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial \mathbf{a}(\mathbf{x},\mathbf{u},t)}{\partial \mathbf{x}}\right)^T\cdot\boldsymbol{\lambda} - \left(\frac{\partial g(\mathbf{x},\mathbf{u},t)}{\partial \mathbf{x}}\right)^T \qquad \text{Eq. 2.31}$$

for any $t \in [t_0, t_f]$, where $\mathbf{u}(t)$ is determined by q algebraic equations:

$$\left(\frac{\partial \mathbf{a}(\mathbf{x},\mathbf{u},t)}{\partial \mathbf{u}}\right)^T \cdot \boldsymbol{\lambda} + \left(\frac{\partial g(\mathbf{x},\mathbf{u},t)}{\partial \mathbf{u}}\right)^T = \mathbf{0} \qquad \text{Eq. 2.32}$$

The boundary conditions for Eqs. 2.30 and 2.31 are split. That is, some are specified for $t = t_0$ and some are specified for $t = t_f$:

$$\mathbf{x}(t_0) = \mathbf{x}_0 \qquad \text{Eq. 2.33}$$

$$\boldsymbol{\lambda}(t_f) = \left(\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}(t_f))\right)^T \qquad \text{Eq. 2.34}$$

Thus, we need to find a solution for a non-linear, two-point boundary-value problem.

2.2.2. Optimal Control problems with control boundaries

Realistic Optimal Control problems involve control limitations. For example, the thrust of a rocket engine may not exceed a certain value, or the driving torque available for a road vehicle is limited by the maximum engine output. In this paragraph we shall extend the above framework for dealing with Optimal Control problems with control constraints. The generalisation of the necessary conditions for optimality leads to the Pontryagin's Minimum Principle. The derivation which is given here for the Minimum Principle, though, is purely heuristic. For a rigorous proof the reader is referred to (Pontryagin et al., 1962).

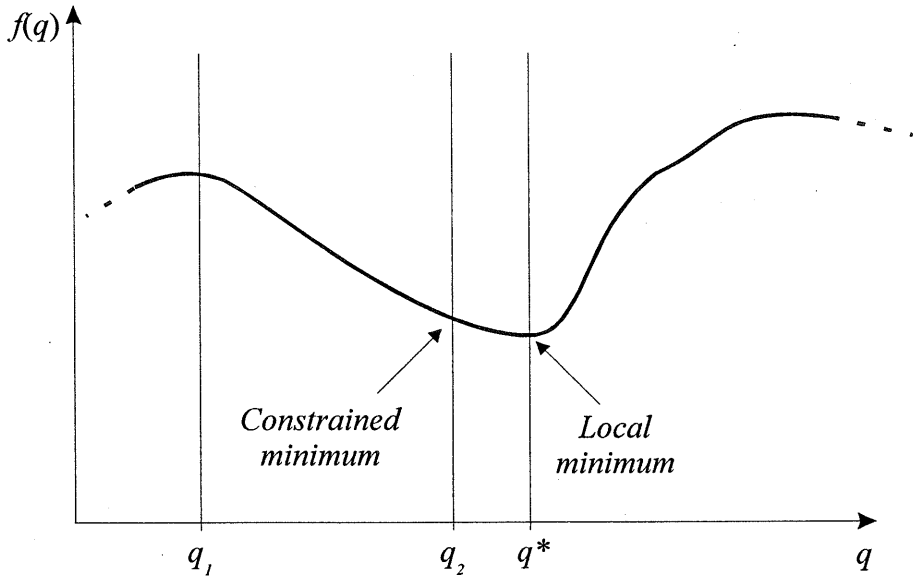


Figure 2.2 Constrained minimum of a generic function.

To begin with, let us analyse the analogous case in calculus. Consider a continuous and differentiable function $f(q)$, as the one qualitatively shown in figure 2.2. If we do not consider any restriction for the values that q may assume, we see that this function has a local minimum in $q = q^*$. Here, the necessary condition that the derivative of $f(q)$ vanishes at the extremum applies. If we then restrict the value of q

within the interval $[q_1, q_2]$, we see that the function $f(q)$ has a minimum point in this interval when $q = q_2$, but here the above necessary condition does not apply. Instead, the necessary conditions for $f(q)$ to have relative minima at the end points of the interval are as follows. If we consider the linear part of the increment of $f(q)$, i.e. the differential of $f(q)$, such increment must always be positive for any *admissible* variation Δq:

$$\Delta f(q_1, \Delta q) \cong \left(\frac{df(q_1)}{dq}\right) \cdot \Delta q \geq 0 \qquad \forall \Delta q \geq 0$$
$$\Delta f(q_2, \Delta q) \cong \left(\frac{df(q_2)}{dq}\right) \cdot \Delta q \geq 0 \qquad \forall \Delta q \leq 0 \qquad \text{Eq. 2.35}$$

In other words, when $q = q_1$ we are only allowed to increase q, and the condition for this point to be a local minimum is that the differential of $f(q)$ must be zero or positive. Instead, when $q = q_2$, we are only allowed to decrease q, and the condition for this point to be a local minimum is again that the differential of $f(q)$ must be zero or positive.

If we apply the same idea to the corresponding problem of Calculus of Variations, the condition stated in Eq. 2.28 changes as follows:

$$\delta \bar{J}(\mathbf{u}, \delta \mathbf{u}) = \int_{t_0}^{t_f} \left[\left(\frac{\partial H}{\partial \mathbf{u}} \right) \cdot \delta \mathbf{u} \right] dt \geq 0 \qquad \text{Eq. 2.36}$$

Here, $\delta \mathbf{u}$ must be admissible. That is, the control $\mathbf{u} + \delta \mathbf{u}$ must not violate the control constraints. If we now expand the integrand in Eq. 2.36 by writing explicitly the variation of the Hamiltonian, we obtain:

$$\int_{t_0}^{t_f} [H(\mathbf{x}, \mathbf{u} + \delta \mathbf{u}, \boldsymbol{\lambda}, t) - H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t)] dt \geq 0 \qquad \text{Eq. 2.37}$$

Eq. 2.37 is satisfied only if:

$$H(\mathbf{x}, \mathbf{u} + \delta \mathbf{u}, \boldsymbol{\lambda}, t) \geq H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) \qquad \forall \textit{ admissible } \delta \mathbf{u} \qquad \text{Eq. 2.38}$$

The condition that an optimal control must minimise the Hamiltonian is called the Pontryagin's Minimum Principle. Thus, Eq. 2.38 together with Eqs.2.30, 2.31 and the boundary conditions in Eqs. 2.33 and 2.34 constitute the necessary conditions for optimality for the general case of an Optimal Control problem with control constraints.

2.3. INDIRECT METHODS FOR SOLVING OPTIMAL CONTROL PROBLEMS

Indirect methods aim to solve an Optimal Control problem by applying the optimality conditions explicitly. This involves the setting up of the adjoint equations 2.31 and 2.34, and the optimality condition 2.32, and requires an iterative procedure to solve the resulting non-linear, two-point, boundary value problem. The general approach consists

of using an initial guess to obtain a solution to a problem where one or more of the optimality conditions is not satisfied. This solution is then used to adjust the initial guess in order to force the next solution to be closer to satisfying all the necessary conditions, until the iterative procedure eventually converges. To illustrate this approach, a brief description of the *Steepest Descent* method (or *Gradient* method), which may be regarded as the simplest among the indirect methods, is given (Kirk, 1970).

Let us consider an Optimal Control problem without state and control constraints. Suppose that a nominal control history $\mathbf{u}^{(i)}$ is known and is used to solve the state differential equations from t_0 to t_f, with initial conditions defined by Eq. 2.33. By using the final values of the state trajectories, Eq. 2.34 may be solved and the co-state equations may then be integrated backwards, from t_f to t_0. The nominal state and the co-state trajectories will therefore satisfy the boundary conditions. If Eq. 2.32 is also satisfied, the control history $\mathbf{u}^{(i)}$ is optimal. If this is not the case, we may evaluate the variation of the performance measure as a result of a variation of the control history by using Eq. 2.28:

$$\delta \bar{J}(\mathbf{u}, \delta \mathbf{u}) = \int_{t_0}^{t_f} \left[\left(\frac{\partial H(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \boldsymbol{\lambda}^{(i)}, t)}{\partial \mathbf{u}} \right) \cdot \delta \mathbf{u} \right] dt \qquad \text{Eq. 2.39}$$

The variation of a functional is the linear part of its increment (see Eq. 2.14). Therefore if we take $\delta \mathbf{u}$ sufficiently small we may assume that the sign of the increment $\Delta \bar{J}$ will be determined by the sign of its variation. As our goal is to minimise the performance measure, we wish to make such increment negative. Hence, we may select a new improved control history as:

$$\delta \mathbf{u} = \mathbf{u}^{(i+1)} - \mathbf{u}^{(i)} = -\tau \cdot \left(\frac{\partial H^{(i)}}{\partial \mathbf{u}} \right)^T \qquad \text{Eq. 2.40}$$

Substituting Eq. 2.40 in Eq. 2.39 yields:

$$\delta \bar{J}(\mathbf{u}, \delta \mathbf{u}) = -\tau \int_{t_0}^{t_f} \left(\frac{\partial H^{(i)}}{\partial \mathbf{u}} \right) \cdot \left(\frac{\partial H^{(i)}}{\partial \mathbf{u}} \right)^T dt \leq 0 \qquad \text{Eq. 2.41}$$

Because the integrand is positive for all t, the variation of the functional will be negative, unless the control is optimal, in which case the equality sign holds and:

$$\frac{\partial H^{(i)}}{\partial \mathbf{u}} = \mathbf{0} \qquad t \in [t_0, t_f] \qquad \text{Eq. 2.42}$$

Choosing the parameter τ sufficiently small ensures that the performance measure decreases at any iteration. The iterative procedure is continued until a suitable convergence criterion is satisfied. The Steepest Descent method is very simple to

implement and is very robust with respect to an inaccurate initial guess of the control history. However it is also the least efficient among the indirect methods, especially near the solution, as it only uses first order sensitivity information.

If the Hamiltonian is a quadratic function of the control history, Eq. 2.32 may be solved explicitly with respect to **u** and the result substituted in the state and co-state equations to obtain the following reduced differential equations, where the dependency on the control input has been eliminated:

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x},\boldsymbol{\lambda},t) \qquad \text{Eq. 2.43}$$

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial H(\mathbf{x},\boldsymbol{\lambda},t)}{\partial \mathbf{x}}\right)^T = -\left(\frac{\partial \mathbf{a}(\mathbf{x},\boldsymbol{\lambda},t)}{\partial \mathbf{x}}\right)^T \cdot \boldsymbol{\lambda} - \left(\frac{\partial g(\mathbf{x},\boldsymbol{\lambda},t)}{\partial \mathbf{x}}\right)^T \qquad \text{Eq. 2.44}$$

A *Quasi-Linearisation* algorithm (Kirk, 1970, Bryson and Ho, 1975) may then be implemented to solve the two-point boundary value problem. This involves expanding the differential equations in a Taylor series about a nominal trajectory $\mathbf{x}^{(i)}$, $\boldsymbol{\lambda}^{(i)}$, and retaining only the terms up to first order. The linearised problem reads:

$$\dot{\mathbf{x}}^{(i+1)} = \mathbf{a}(\mathbf{x}^{(i)},\boldsymbol{\lambda}^{(i)},t) + \left[\frac{\partial \mathbf{a}(\mathbf{x}^{(i)},\boldsymbol{\lambda}^{(i)},t)}{\partial \mathbf{x}}\right]\left[\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}\right] + \left[\frac{\partial \mathbf{a}(\mathbf{x}^{(i)},\boldsymbol{\lambda}^{(i)},t)}{\partial \boldsymbol{\lambda}}\right]\left[\boldsymbol{\lambda}^{(i+1)} - \boldsymbol{\lambda}^{(i)}\right] \qquad \text{Eq. 2.45}$$

$$\dot{\boldsymbol{\lambda}}^{(i+1)} = -\left(\frac{\partial H(\mathbf{x}^{(i)},\boldsymbol{\lambda}^{(i)},t)}{\partial \mathbf{x}}\right)^T - \left[\frac{\partial^2 H(\mathbf{x}^{(i)},\boldsymbol{\lambda}^{(i)},t)}{\partial \mathbf{x}^2}\right]\left[\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}\right] - \left[\frac{\partial^2 H(\mathbf{x}^{(i)},\boldsymbol{\lambda}^{(i)},t)}{\partial \mathbf{x}\partial \boldsymbol{\lambda}}\right]\left[\boldsymbol{\lambda}^{(i+1)} - \boldsymbol{\lambda}^{(i)}\right] \qquad \text{Eq. 2.46}$$

The advantage in doing this is that a linear two-point boundary value problem is readily solved by applying the superposition principle. The solution for the original non-linear problem may then be found by solving a sequence of linearised problems starting from an initial estimate for the state and the co-state trajectories. Such estimate is not required to meet the boundary conditions defined in Eqs. 2.33 and 2.34. After the first iteration, though, all the intermediate solutions will do so. Under the proper conditions, the Quasi-Linearisation method converges quadratically when near the optimum. The implementation of this method is however very difficult because of the differentiation required to obtain the linearised equations. Furthermore, the initial estimate of the solution must be sufficiently accurate. Otherwise the algorithm will not converge.

A simpler approach to solving the non-linear, two-point boundary value problem defined by the reduced differential equations 2.43 and 2.44 is the *shooting* method (Press et al., 1992). In this case the boundary value problem is treated as an initial value problem by estimating the free initial conditions. The differential equations are integrated forward until the end of the integration interval where, in general, there will be some discrepancies with the desired boundary values. The free initial conditions are

then adjusted in order to reduce such discrepancies, until they eventually meet a suitable tolerance criterion. This is usually treated as a multidimensional root finding problem and is solved by using a Newton-Raphson method. A shooting method is simple to implement and efficient. The main problem here, though, is to find a good initial estimate for the free initial conditions, otherwise the method will fail to converge.

So far we have restricted the discussion to the solution of unconstrained Optimal Control problems, and the task of solving the non-linear two-point boundary value problem already appears considerably difficult. However, realistic Optimal Control problems often involve complex state and control algebraic constraints and this is a serious obstacle for the application of any of the methods described so far. A common practice is to include all the algebraic constraints into the performance measure by means of penalty functions. This restores the possibility to use any of the above strategies, but may result in an ill-conditioned numerical problem. Choosing the penalty parameters will then be crucial to ensure that the solution of the unconstrained problem with penalty functions converges to that of the constrained problem. Another possibility to deal with constrained Optimal Control problems is to use the *Sequential Conjugate Gradient Restoration* algorithm (Cloutier, Mohanty and Miele, 1977, Wu and Miele, 1978a and 1978b). In this method each iteration is split in two phases. One phase is meant to satisfy the algebraic constraints, while the second phase is intended to decrease the performance measure. Hence, all the iterations are feasible, that is they satisfy the constraints. The property of this method is that it is compatible with many practical optimisation problems which involve general state and control constraints and it does not require a very accurate initial trial solution. Its efficiency, though, is comparable with that of the ordinary gradient method and it improves only when specifying tight tolerances, as the conjugate gradient methods behave more like a second order method when near the solution.

A class of numerical methods for solving boundary value problems, known as *Relaxation* methods (or *Collocation* methods) (Press et al., 1992, Ascher, Christiansen and Russell, 1981) offers the best possibility to solve an Optimal Control problem with general state and control constraints. The differential equations are replaced by finite-difference equations on a mesh of points that covers the range of the integration. A trial solution consists of values for the dependent variables, i.e. the state and the co-state variables, at each mesh point. Such trial solution is not required to satisfy the finite-difference equations and not even the boundary conditions. An iterative procedure is then started, whose task is to adjust all the dependent variables in order to bring them into successively closer agreement with the finite-difference equations and simultaneously with the boundary conditions. General state and control constraint equations assume the form of internal boundary conditions that the dependent variables must satisfy at each point of the mesh where such constraints are active. Despite the large number of dependent variables, Relaxation methods are very efficient unless the solution is very oscillatory, in which case the number of mesh points would be too large. Also, they allow to specify complex boundary conditions which cannot be solved in closed form. The difficulty here is, though, to provide a sufficiently accurate initial trial solution.

Although robust and efficient algorithms for solving two-point boundary value problems are available, the use of indirect methods for solving Optimal Control problems poses many difficulties. First of all, deriving the adjoint equations 2.31 and the optimality condition 2.32 involves the symbolic differentiation of the problem

equations, which may become a very difficult task as the complexity of the mathematical model of the plant grows. Ultimately, it may not be possible if, for example, look up tables or other interpolation techniques are used within the modelling process. Some authors have proposed the use of Automatic Differentiation techniques in order to evaluate the co-state derivatives required to integrate Eq. 2.31 without actually deriving symbolic expressions for the co-state equations (Mehlhorn, Dinkelmann and Sachs, 1994). The other major problem lies in the fact that most of the methods described here require an accurate initial estimate of the solution. While the knowledge of the physical properties of the plant allows to find a reasonable estimate of the initial state trajectory and control history, no information is available to guess the initial co-state trajectory. A solution to this problem is often to start the optimisation with a low order method, like the Steepest Descent method, until a reasonable estimate of the solution is obtained. Then, the solution is refined by using a more efficient method. Finally, another important aspect characterising the methods described so far, is that they assume the control history to be continuous. If the control input is discontinuous or anyway rapidly varying, as is the case for the longitudinal vehicle control when it switches from full driving into braking, the general approach should be modified by dividing the trajectory in segments which would have to be joined at the points where the control switches occur, ensuring also the continuity of the state trajectory. Needless to say, this provides an extra amount of complexity for solving the Optimal Control problem.

2.4. DIRECT METHODS FOR SOLVING OPTIMAL CONTROL PROBLEMS

A class of methods for solving Optimal Control problems, known as *direct transcription methods*, does not use the necessary conditions for optimality and the Pontryagin's Minimum Principle. Instead, the original Optimal Control problem is converted into a *Non-Linear Programming* problem and is solved directly using mathematical programming techniques. Direct methods for the solution of Optimal Control problems are nowadays widely used in practical applications, e.g. in aerospace trajectory optimisation. They are extensively described in the literature of computational optimal control (Betts, 1994, Enright and Conway, 1990, Kraft, 1994, Kraft and Bulirsch, 1994, Stryk, 1993).

The basic concept of direct methods is that the continuous control history is replaced with a discrete approximation. It is assumed that the control input can only be adjusted at a number of fixed positions along the trajectory, while the intermediate values are estimated by means of interpolation techniques. Let $\mathbf{u}_n$ be the vector of discrete control parameters and $\mathbf{t}_n$ be the vector of the corresponding instances within the time interval $[t_0, t_f]$. The actual control input to the plant at any time t is evaluated as:

$$\mathbf{u}(t) = \text{interp}(\mathbf{t}_n, \mathbf{u}_n, t) \qquad \text{Eq. 2.47}$$

The control parameters univocally determine the control history which, in turn, determine the system state trajectory. Therefore the performance measure and the constraints may be expressed directly as functions of these control parameters. Hence, the original Optimal Control problem may be stated as a Non-Linear Programming

problem, that is, to find the set of control parameters $\mathbf{u}_n$ which minimises a generic non-linear multi-variable function subjected to general equality and inequality constraints:

$$\begin{aligned} &\min_{\mathbf{u}_n} \quad J(\mathbf{u}_n) \\ &\textit{subject to}: \quad c_i(\mathbf{u}_n) = 0 \qquad i \in E \\ &\qquad\qquad\quad c_i(\mathbf{u}_n) \le 0 \qquad i \in I \end{aligned} \qquad \text{Eq. 2.48}$$

Here, E and I represent the set of equality and inequality constraints respectively. The performance measure $J(\mathbf{u}_n)$ is often referred to in the context of numerical optimisation simply as the *objective function.*

The first order, necessary conditions for the set of independent variables $\mathbf{u}_n$ to be a constrained minimiser for the function $J(\mathbf{u}_n)$ are known as the *Kuhn-Tucker* conditions, and a solution for the problem defined by Eq. 2.48 is often referred to as a *KT point.* In the next paragraph we shall describe the Kuhn-Tucker conditions, but we shall also extend the theoretical background to consider second order, sufficient conditions for optimality. These are important in order to successively describe the *Newton-Lagrange* method (*Sequential Quadratic Programming* method, or simply *SQP*), which is nowadays the most powerful method to solve constrained optimisation problems. The theory of numerical optimisation, though, is presented here focusing mainly on the practical aspects involved. For a rigorous development of the subject the reader may refer to (Fletcher, 1987, Gill et al., 1981). Finally, different strategies may be employed to convert an Optimal Control problem into a Non-Linear Programming problem and these will be described at the end of this paragraph.

2.4.1. The Kuhn-Tucker conditions

Consider firstly a constrained minimisation problem with only equality constraints:

$$\begin{aligned} &\min_{\mathbf{u}_n} \quad J(\mathbf{u}_n) \\ &\textit{subject to}: \quad c_i(\mathbf{u}_n) = 0 \qquad i \in E \end{aligned} \qquad \text{Eq. 2.49}$$

Let us adjoin the constraints to the objective function by using the Lagrange multipliers $\boldsymbol{\lambda}$ to form the *Lagrangian* function:

$$L(\mathbf{u}_n, \boldsymbol{\lambda}) = J(\mathbf{u}_n) - \sum_{i \in E} \lambda_i \cdot c_i(\mathbf{u}_n) = J(\mathbf{u}_n) - \boldsymbol{\lambda}^T \cdot \mathbf{c}(\mathbf{u}_n) \qquad \text{Eq. 2.50}$$

The first order, necessary condition for a local minimiser for the problem defined by Eq. 2.49 is that $\mathbf{u}_n^*$, $\boldsymbol{\lambda}^*$ is a stationary point for the Lagrangian function:

$$\begin{cases} \dfrac{\partial L(\mathbf{u}_n^*, \boldsymbol{\lambda}^*)}{\partial \mathbf{u}_n} = \dfrac{\partial J(\mathbf{u}_n^*)}{\partial \mathbf{u}_n} - \boldsymbol{\lambda}^T \cdot \dfrac{\partial \mathbf{c}(\mathbf{u}_n^*)}{\partial \mathbf{u}_n} = \mathbf{0} \\ \dfrac{\partial L(\mathbf{u}_n^*, \boldsymbol{\lambda}^*)}{\partial \boldsymbol{\lambda}} = -\mathbf{c}(\mathbf{u}_n^*) = \mathbf{0} \end{cases} \qquad \text{Eq. 2.51}$$

The second condition in Eq. 2.51 simply states the feasibility of the local minimiser, while the first condition states that the gradient of the objective function must be a linear combination of the rows of the Jacobian of the constraints. This condition is qualitatively described in figure 2.3 by considering a two-dimensional function $J(\mathbf{u}_n)$ with only one equality constraint. For a generic feasible point along the constraint $c(\mathbf{u}_n)$ we may always find a feasible direction $\mathbf{d}$ for which both of the two following conditions hold:

$$\begin{cases} \dfrac{\partial c}{\partial \mathbf{u}_n}\cdot\mathbf{d}=0 & \Rightarrow \textit{feasible direction} \\ \dfrac{\partial J}{\partial \mathbf{u}_n}\cdot\mathbf{d}<0 & \Rightarrow \textit{descent direction} \end{cases} \qquad \text{Eq. 2.52}$$

However, if a point $\mathbf{u}_n^*$ is a local minimiser, there can not be a feasible direction departing from it which is also a descent direction. This, though, can only be if the direction of the gradient of the objective is the same as the direction of the gradient of the constraints, that is:

$$\frac{\partial J(\mathbf{u}_n^*)}{\partial \mathbf{u}_n}=\lambda\cdot\frac{\partial c(\mathbf{u}_n^*)}{\partial \mathbf{u}_n} \qquad \text{Eq. 2.53}$$

Thus, Eq. 2.51 is a necessary condition for a constrained local minimiser to occur and it means that any descent direction points towards violating the constraints.

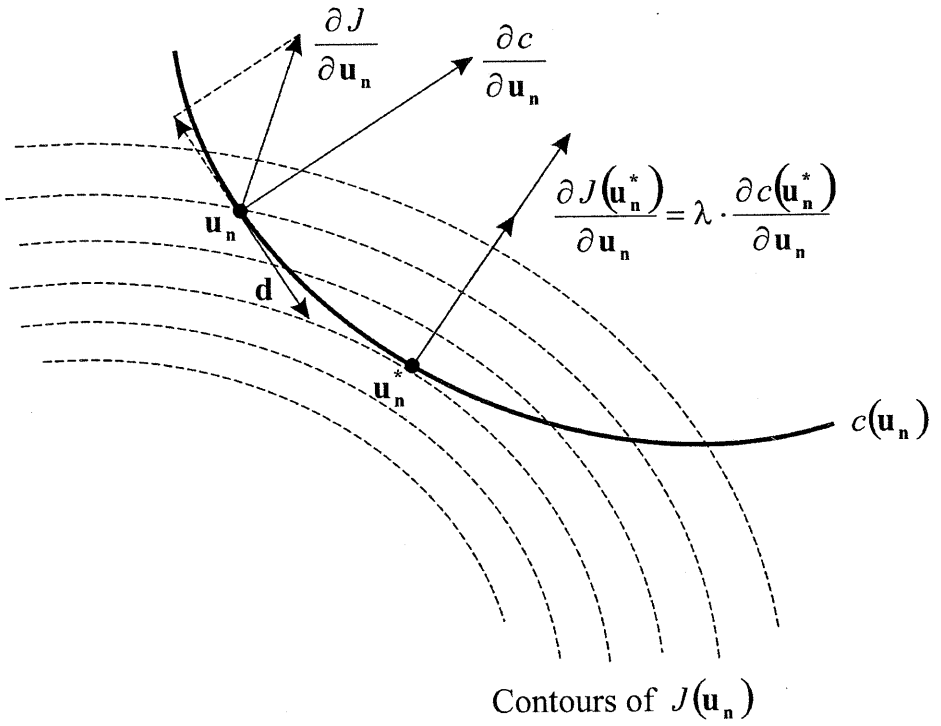


Figure 2.3 Existence of the Lagrange multiplier at a local constrained minimum.

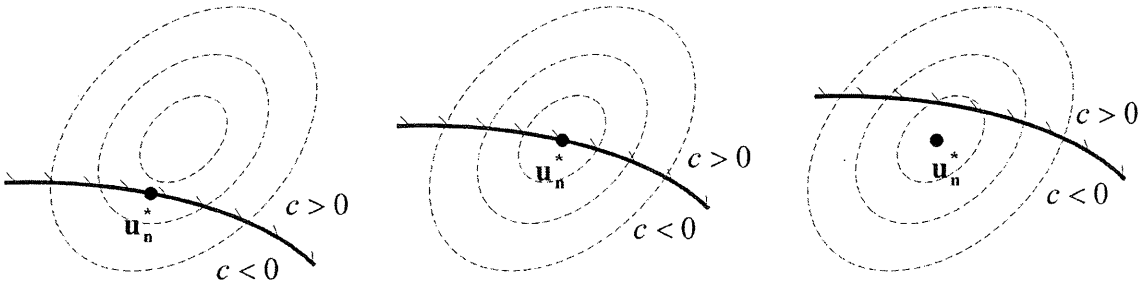
Let us now consider the full problem involving also inequality constraints. Firstly, we may observe that only the active constraints at the solution matter (we shall define the set A of active constraints as a sub-set of the inequality constraints I: $A \subseteq I$). A feasible direction at the solution is then defined as:

$$\frac{\partial c_i(\mathbf{u}_n^*)}{\partial \mathbf{u}_n} \cdot \mathbf{d} \leq 0 \qquad \forall i \in A \qquad \text{Eq. 2.54}$$

This means that from the point $\mathbf{u}_n^*$ we are allowed to move towards the region where the constraints assume non-positive values. Therefore, for $\mathbf{u}_n^*$ to be a local minimiser for the problem defined by Eq. 2.48, there must not be any descent direction which points towards the feasible region. This translates into a further necessary condition on the sign of the Lagrange multipliers for the active inequality constraints, that is:

$$\lambda_i \leq 0 \qquad \forall i \in A \qquad \text{Eq. 2.55}$$

Note that the sign of the Lagrange multipliers depends on how the inequality constraints are defined, i.e. whether the feasible region is where the constraints assume non-positive or non-negative values.



a) Strongly active $\lambda_i^* < 0, \quad c_i^* = 0$

b) Weakly active $\lambda_i^* = c_i^* = 0$

c) Inactive $\lambda_i^* = 0, \quad c_i^* < 0$

Figure 2.4 Complementary condition.

Summarising, the full set of first order necessary conditions for a point $\mathbf{u}_n^*$ to be a local constrained minimum for the problem defined by Eq. 2.48 reads:

$$\begin{aligned}
&\frac{\partial L(\mathbf{u}_n^*, \boldsymbol{\lambda}^*)}{\partial \mathbf{u}_n} = 0 \\
&c_i(\mathbf{u}_n^*) = 0 \qquad i \in E \\
&c_i(\mathbf{u}_n^*) \leq 0 \qquad i \in I \\
&\lambda_i \leq 0 \qquad i \in I \\
&\lambda_i \cdot c_i(\mathbf{u}_n^*) = 0 \qquad \forall i
\end{aligned} \qquad \text{Eq. 2.56}$$

These are the Kuhn-Tucker conditions. The final condition in Eqs. 2.56 is referred to as the *complementary condition* and states that both λ_i and c_i cannot be non-zero at the same time, or equivalently that inactive constraints have zero multipliers (see figure 2.4). A particular condition is when the constraint is weakly active, that is the case in which the constraint boundary lies where the unconstrained minimum would be. If this does not happen, *strict complementarity* is said to hold.

Let us now consider second order necessary and sufficient conditions for optimality, which give information about the curvature of the objective and constraints functions at a local minimiser. To begin with, we shall again consider only problems with equality constraints. Let us assume that $\mathbf{u_n}^*$, λ^* is a KT point for the problem defined by Eq. 2.49. It then follows that:

$$J(\mathbf{u_n}^*) = L(\mathbf{u_n}^*, \lambda^*) \qquad \text{Eq. 2.57}$$

since all the constraints must be zero. Consider then an incremental step $\delta\mathbf{u_n}$ along a generic feasible direction $\mathbf{d}$, defined as:

$$\frac{\partial \mathbf{c}(\mathbf{u_n}^*)}{\partial \mathbf{u_n}} \cdot \mathbf{d} = \mathbf{0} \qquad \text{Eq. 2.58}$$

and consider the Taylor expansion of $J(\mathbf{u_n}^*)$ about the KT point:

$$J(\mathbf{u_n}^* + \delta\mathbf{u_n}) = L(\mathbf{u_n}^* + \delta\mathbf{u_n}, \lambda^*) = L(\mathbf{u_n}^*, \lambda^*) + \left(\frac{\partial L(\mathbf{u_n}^*, \lambda^*)}{\partial \mathbf{u_n}}\right) \cdot \delta\mathbf{u_n} + \frac{1}{2} \cdot \delta\mathbf{u_n}^T \cdot \mathbf{H} \cdot \delta\mathbf{u_n} + o\left(\delta\mathbf{u_n}^T \cdot \delta\mathbf{u_n}\right) \qquad \text{Eq. 2.59}$$

Here, $\mathbf{H}$ is the *Hessian* matrix with respect to $\mathbf{u_n}$ of the Lagrangian function:

$$\mathbf{H} = \frac{\partial^2 L}{\partial \mathbf{u_n}^2} = \frac{\partial^2 J}{\partial \mathbf{u_n}^2} - \sum_{i \in E} \lambda_i \cdot \frac{\partial^2 c_i}{\partial \mathbf{u_n}^2} \qquad \text{Eq. 2.60}$$

Since $\mathbf{u_n}^*$, λ^* is a KT point, the linear term in Eq. 2.59 is equal to 0. Then, by using Eq. 2.57, Eq. 2.59 reduces to:

$$J(\mathbf{u_n}^* + \delta\mathbf{u_n}) - J(\mathbf{u_n}^*) = \frac{1}{2} \cdot \delta\mathbf{u_n}^T \cdot \mathbf{H} \cdot \delta\mathbf{u_n} + o\left(\delta\mathbf{u_n}^T \cdot \delta\mathbf{u_n}\right) \qquad \text{Eq. 2.61}$$

If $\mathbf{u_n}^*$ is a local minimiser, the right hand term of Eq. 2.61 must be non-negative. By taking the limit of Eq. 2.61 for $\delta\mathbf{u_n}$ which tends to 0, it follows that:

$$\mathbf{d}^T \cdot \mathbf{H} \cdot \mathbf{d} \geq 0 \qquad \text{Eq. 2.62}$$

Thus a second order necessary condition for a local minimiser is that the Lagrangian function must have a non-negative curvature for all feasible directions at $\mathbf{u_n}^*$. Eq. 2.62

becomes a sufficient condition for a strict local minimiser simply by taking only the inequality:

$$\mathbf{d}^T \cdot \mathbf{H} \cdot \mathbf{d} > 0 \qquad \text{Eq. 2.63}$$

Finally, for a problem with inequality constraints, the same conditions apply when the set of active constraints is considered.

2.4.2. The Newton-Lagrange method (SQP algorithm)

The Newton-Lagrange method may be interpreted as a Newton method applied to find a stationary point for the Lagrangian function. This involves the minimisation of a quadratic model of the constrained minimisation problem in order to obtain an improved solution, and to iterate until the procedure eventually converges to the minimum point. To outline the method, let us consider once again a problem involving only equality constraints, i.e. the problem defined by Eq. 2.49. Let us firstly define the derivative operator ∇ for the Lagrangian function as:

$$\nabla = \begin{bmatrix} \dfrac{\partial}{\partial \mathbf{u}_n} \\ \dfrac{\partial}{\partial \boldsymbol{\lambda}} \end{bmatrix} \qquad \text{Eq. 2.64}$$

so that Eq. 2.51 may simply be stated as follows:

$$\nabla L(\mathbf{u}_n^*, \boldsymbol{\lambda}^*) = 0 \qquad \text{Eq. 2.65}$$

Let $\mathbf{u}_n^{(i)}$ and $\boldsymbol{\lambda}^{(i)}$ be a trial solution. The Taylor series for ∇L about $\mathbf{u}_n^{(i)}$, $\boldsymbol{\lambda}^{(i)}$ reads:

$$\nabla L\left(\mathbf{u}_n^{(i)} + \delta\mathbf{u}_n, \boldsymbol{\lambda}^{(i)} + \delta\boldsymbol{\lambda}\right) = \nabla L\left(\mathbf{u}_n^{(i)}, \boldsymbol{\lambda}^{(i)}\right) + \left[\nabla^2 L\left(\mathbf{u}_n^{(i)}, \boldsymbol{\lambda}^{(i)}\right)\right] \cdot \begin{bmatrix} \delta\mathbf{u}_n \\ \delta\boldsymbol{\lambda} \end{bmatrix} + \ldots \qquad \text{Eq. 2.66}$$

Neglecting higher order terms and setting the left hand side term of Eq. 2.66 to zero by virtue of Eq. 2.65, gives the following iteration:

$$\left[\nabla^2 L\left(\mathbf{u}_n^{(i)}, \boldsymbol{\lambda}^{(i)}\right)\right] \cdot \begin{bmatrix} \delta\mathbf{u}_n \\ \delta\boldsymbol{\lambda} \end{bmatrix} = -\nabla L\left(\mathbf{u}_n^{(i)}, \boldsymbol{\lambda}^{(i)}\right) \qquad \text{Eq. 2.67}$$

Expanding the derivative operator and using Eqs. 2.51 and 2.60 yields:

$$\begin{bmatrix} \mathbf{H}^{(i)} & \left(\dfrac{\partial \mathbf{c}^{(i)}}{\partial \mathbf{u}_n}\right)^T \\ \left(\dfrac{\partial \mathbf{c}^{(i)}}{\partial \mathbf{u}_n}\right) & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \delta\mathbf{u}_n \\ \delta\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\left(\dfrac{\partial J^{(i)}}{\partial \mathbf{u}_n}\right)^T + \left(\dfrac{\partial \mathbf{c}^{(i)}}{\partial \mathbf{u}_n}\right)^T \cdot \boldsymbol{\lambda}^{(i)} \\ \mathbf{c}^{(i)} \end{bmatrix} \qquad \text{Eq. 2.68}$$

Let us now apply the following definitions:

$$\begin{aligned}\boldsymbol{\delta}^{(i)} &= \delta\mathbf{u}_n \\ \boldsymbol{\lambda}^{(i+1)} &= \boldsymbol{\lambda}^{(i)} + \delta\boldsymbol{\lambda}\end{aligned} \qquad \text{Eq. 2.69}$$

By using Eq. 2.69 into Eq. 2.68 we obtain:

$$\begin{bmatrix} \mathbf{H}^{(i)} & \left(\frac{\partial \mathbf{c}^{(i)}}{\partial \mathbf{u}_n}\right)^T \\ \left(\frac{\partial \mathbf{c}^{(i)}}{\partial \mathbf{u}_n}\right) & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\delta}^{(i)} \\ \boldsymbol{\lambda}^{(i+1)} \end{bmatrix} = \begin{bmatrix} -\left(\frac{\partial J^{(i)}}{\partial \mathbf{u}_n}\right)^T \\ \mathbf{c}^{(i)} \end{bmatrix} \qquad \text{Eq. 2.70}$$

The solution of Eq. 2.70 yields a correction $\boldsymbol{\delta}^{(i)}$ for the independent variables and an improved set of Lagrange multipliers. Finally:

$$\mathbf{u}_n^{(i+1)} = \mathbf{u}_n^{(i)} + \boldsymbol{\delta}^{(i)} \qquad \text{Eq. 2.71}$$

It is possible to restate this method in terms of one in which the minimisation of a *Quadratic Programming* sub-problem is involved. Such a sub-problem reads:

$$\begin{aligned} &\min_{\boldsymbol{\delta}} \quad \left(q^{(i)}(\boldsymbol{\delta})\right) \\ &\textit{subject to} \quad \mathbf{l}^{(i)}(\boldsymbol{\delta}) = \mathbf{0} \\ &\textit{and / or} \quad \mathbf{l}^{(i)}(\boldsymbol{\delta}) \le \mathbf{0} \end{aligned} \qquad \text{Eq. 2.72}$$

where:

$$\begin{aligned} q^{(i)}(\boldsymbol{\delta}) &= \frac{1}{2}\cdot\boldsymbol{\delta}^T\cdot\mathbf{H}\cdot\boldsymbol{\delta} + \frac{\partial J^{(i)}}{\partial \mathbf{u}_n}\cdot\boldsymbol{\delta} + J^{(i)} \\ \mathbf{l}^{(i)}(\boldsymbol{\delta}) &= \frac{\partial \mathbf{c}^{(i)}}{\partial \mathbf{u}_n}\cdot\boldsymbol{\delta} + \mathbf{c}^{(i)} \end{aligned} \qquad \text{Eq. 2.73}$$

This sub-problem bears a nice relationship with the original non-linear constrained optimisation problem of Eq. 2.48. The non-linear constraints are replaced by their first order Taylor series approximation. Likewise, the objective function is replaced by the quadratic function $q^{(i)}(\boldsymbol{\delta})$. This is a second order Taylor series approximation of the objective function about $\mathbf{u}_n^{(i)}$ but with the addition of constraint curvature terms in the Hessian. This is important in order to ensure second order convergence rate for the non-linear constraints. A *Sequential Quadratic Programming* (SQP) algorithm consists then in solving a sequence of Quadratic Programming sub-problems like that of Eq. 2.72. At each iteration the solution $\boldsymbol{\delta}^{(i)}$ is not used directly to update the independent variables as in Eq. 2.71. Instead, it is used as a *search direction* and a new iterate is formed as follows:

$$\mathbf{u}_n^{(i+1)} = \mathbf{u}_n^{(i)} + \tau \cdot \boldsymbol{\delta}^{(i)} \qquad \text{Eq. 2.74}$$

In order to quantify the improvement towards the solution to the optimisation problem, a *merit function* is defined. The merit function combines the objective function and the constraints into a single measure:

$$\phi(\mathbf{u}_n) = \phi(J(\mathbf{u}_n), \mathbf{c}(\mathbf{u}_n)) \qquad \text{Eq. 2.75}$$

The task is then to find a value for the parameter τ which produces a sufficient decrease in the merit function. This constitute a one-dimensional minimisation problem which is solved with a suitable *line search* algorithm:

$$\min_{\tau} \quad \phi\left(\mathbf{u}_n^{(i)} + \tau \cdot \boldsymbol{\delta}^{(i)}\right) \qquad \text{Eq. 2.76}$$

Practical implementations of the SQP algorithm do not require the Hessian matrix explicitly, as the second order derivatives would be either very complex to derive symbolically, or computationally too expensive and very difficult to obtain by means of numerical differentiation techniques with sufficient accuracy. Instead, numerical updating techniques, like the BFGS formula (Fletcher, 1987) are implemented in order to build up curvature information as the iterations proceed. As a consequence the efficiency of the algorithm is reduced, but this is generally regarded as acceptable compared to the effort that it would take to compute the Hessian directly.

The various existing implementations of the SQP algorithm differ in many details, e.g. the strategy used in the line search algorithms, the definition of the merit function, the numerical methods implemented in order to ensure a robust behaviour in case the Hessian approximation fails to be positive definite, etc. We shall not consider these aspects, though, as the matter is regarded as too specific for the purpose of this review. More details may be found in (Fletcher, 1987, Betts, 1993, Gill et al., 1994). We shall, however, give some definitions, which will be useful later on when discussing the strategy for the lap time optimisation program. A first distinction can be made on the basis of how the various algorithms deal with constraint violations. While standard SQP algorithms explicitly allow the intermediate solutions to be infeasible, *feasible SQP* algorithms ensure that no iterations violate the constraints. This is useful when the objective function is not defined outside the feasible region. Finally, the standard SQP algorithms treat the Hessian approximation as a dense matrix. This sets a limit to the size of problems which may be efficiently solved by these algorithms to a maximum of about a thousand independent variables. Instead, *sparse SQP* algorithms take advantage of the sparsity of the Hessian by storing only the non-zero elements. If a problem features a very sparse Hessian, these algorithms can deal with up to several thousands of independent variables.

2.4.3. Direct transcription methods

Direct transcription methods may be divided into different classes depending on how the state differential equations, representing the mathematical model of the plant, are treated. Let us firstly recall the equations which define a general Optimal Control problem. The task is to find an admissible control **u*** which causes a plant to follow a

feasible trajectory **x*** which minimises the performance measure J:

$$\min_{\mathbf{u}} \quad J = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)\,dt$$

$$\begin{aligned} &\textit{subject to}: \quad \dot{\mathbf{x}}(t) = \mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \\ &\qquad\qquad \mathbf{c} \le 0 \\ &\qquad\qquad \mathbf{u}_L \le \mathbf{u}(t) \le \mathbf{u}_U \end{aligned} \qquad \text{Eq. 2.77}$$

$$\textit{for all } t \in [t_0, t_f]$$

First of all, the continuous control history **u** has to be represented by a finite set of control parameters $\mathbf{u_n}$. For each of the q control signals which are returned by the function $\mathbf{u}(t)$, we shall define a time grid as follows:

$$\Delta_{u_i} = \{ t_0 < t_1 < t_2 < \cdots < t_{n_{i-1}} < t_{n_i} = t_f \} \qquad i = 1, \ldots, q \qquad \text{Eq. 2.78}$$

Time grids with different density may be allowed for the various control signals, involving shorter time intervals where the rate of change of the individual controls is expected to be greater. The total number of control parameters will be:

$$n = \sum_{i=1}^{q} n_i \qquad \text{Eq. 2.79}$$

The actual control input to the plant at any time t is evaluated by means of interpolation techniques:

$$\mathbf{u}(t) = \text{interp}(\mathbf{t_n}, \mathbf{u_n}, t) \;\Rightarrow\; \mathbf{u} = \mathbf{u}(\mathbf{u_n}) \qquad \text{Eq. 2.80}$$

Thus, the vector function $\mathbf{u}(t)$ is completely and uniquely defined by the n-dimensional vector $\mathbf{u_n}$:

$$\mathbf{u_n} = \{ \mathbf{u}_{n_1}, \mathbf{u}_{n_2}, \ldots, \mathbf{u}_{n_q} \} \qquad \text{Eq. 2.81}$$

In the *direct shooting method* (Kraft, 1994), also known as *state elimination method*, the state differential equations are satisfied at any iteration by solving them with any suitable method, e.g. Runge-Kutta methods. The control history **u** univocally determines the state trajectory **x**. Therefore, by virtue of Eq. 2.80, we may write:

$$\mathbf{x} = \mathbf{x}(\mathbf{u_n}) \qquad \text{Eq. 2.82}$$

Hence, the performance measure and the constraints may be related directly to the control parameters $\mathbf{u}_n$, and the original Optimal Control problem defined by Eq. 2.77 reduces to the following Non-Linear Programming problem:

$$\begin{aligned} &\min_{\mathbf{u}_n} \quad J(\mathbf{u}_n) \\ &\textit{subject to}: \quad \mathbf{c}(\mathbf{u}_n) \le 0 \\ &\qquad\qquad \mathbf{u}_n^L \le \mathbf{u}_n \le \mathbf{u}_n^U \end{aligned} \qquad \text{Eq. 2.83}$$

This approach is of straightforward application, but it may cause sensitivity problems because of the coupling between the early controls with the later parts of the state trajectories. In other words, early controls will have a much greater effect on the objective and constraint functions, as they influence the whole trajectory. This constitutes a serious problem especially when complex path constraints are present, as is the case for the minimum lap time optimisation problem.

A different approach, known as *direct collocation method* (Betts, 1994, Stryk, 1993, Kraft and Bulirsch, 1994), consists in enforcing the solution of the state differential equations by a collocation procedure, which implies that also the state trajectories are discretised over a fixed grid of points. Let us divide the total time interval $[t_0, t_f]$ into m segments:

$$\Delta_x = \{t_0 < t_1 < t_2 < \cdots < t_{m-1} < t_m = t_f\} \qquad \text{Eq. 2.84}$$

In order to indicate the values of the state variables at each node we shall introduce the following notation:

$$\mathbf{x}_i \equiv \mathbf{x}(t_i) \qquad \text{Eq. 2.85}$$

As we assume that the initial conditions $\mathbf{x}_0$ are fixed, the set of the added independent state parameters is defined as:

$$\mathbf{x}_m = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\} \qquad \text{Eq. 2.86}$$

The state differential equations in Eq. 2.77 may be replaced with a finite difference approximation by introducing the vector of *defects* ξ_i. Different discretisation schemes may be employed for this purpose. The simplest method, known as the *trapezoidal discretisation*, uses the Euler formula:

$$\xi_i = +\mathbf{x}_{i-1} - \mathbf{x}_i + \frac{1}{2}\cdot(\Delta_i)\cdot[\mathbf{a}_i + \mathbf{a}_{i-1}] \qquad i = 1,\ldots m \qquad \text{Eq. 2.87}$$

where $\Delta_i = t_i - t_{i-1}$ and $\mathbf{a}_i = \mathbf{a}(\mathbf{x}_i, \mathbf{u}(t_i), t_i)$. The solution of the state differential equations is then imposed by adding the constraint that all the defects must be zero. Let $\mathbf{y}$ be the vector of independent optimisation variables including both state and control parameters:

$$\mathbf{y} = \mathbf{x}_m \cup \mathbf{u}_n \qquad \text{Eq. 2.88}$$

If there are p state variables, the total number of elements of $\mathbf{y}$ will be:

$$n_{tot} = n + p \times m \qquad \text{Eq. 2.89}$$

Hence, the Non-Linear Programming problem representing the original Optimal Control problem becomes:

$$\begin{aligned} &\min_{\mathbf{u}_n} \quad J(\mathbf{y}) \\ &\textit{subject to}: \quad \mathbf{c}(\mathbf{y}) \le 0 \\ &\qquad\qquad \xi_i(\mathbf{y}) = 0 \qquad i = 1,\ldots m \\ &\qquad\qquad \mathbf{y}_L \le \mathbf{y} \le \mathbf{y}_U \end{aligned} \qquad \text{Eq. 2.90}$$

The direct collocation method ensures maximum decoupling, since each control variable can alter the system states only at neighbouring points. As a consequence, the problem derivatives will also be very sparse. The application of this method results in very large numerical optimisation problems, which typically involve thousands of independent variables and constraints. However, for the right problem and using algorithms which take advantage of the sparsity involved, the method is very efficient.

A third approach, known as the *parallel shooting method*, is the obvious compromise which allows to avoid the sensitivity problems without excessively increasing the problem size (Enright and Conway, 1990). The state trajectories are discretised over time segments which are much longer compared to those required by the direct collocation method. Instead of using the finite difference approximation of Eq. 2.87, within each time segment the state differential equations are solved by using a suitable method, e.g. Runge-Kutta methods. For the i^{th} segment, we then solve the following initial value problem:

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}, \mathbf{u}, t) \qquad \mathbf{x}(t_{i-1}) = \mathbf{x}_{i-1} \qquad t \in [t_{i-1}, t_i] \qquad i = 1,\ldots,m \qquad \text{Eq. 2.91}$$

The continuity of the state trajectories at interior points is enforced by adding the constraint that the defects must be equal to zero at the solution, as for the direct collocation method. In this case, though, the defects are simply defined as the differences between the final state values in the i^{th} segment (obtained by solving problem 2.91) and the initial state values for the successive segment:

$$\xi_i = \mathbf{x}(t_i) - \mathbf{x}_i \qquad i = 1,\ldots,m-1 \qquad \text{Eq. 2.92}$$

Hence, the resulting Non-Linear Programming problem is still defined as in Eq. 2.90.

The decoupling between the early controls and the later parts of the state trajectories is still achieved, as each control variable is effective only within the segment where it occurs. However, the length of the time segments will be crucial in order to avoid the sensitivity problem. The task is then to determine the proper discretisation

scheme which minimises the size of the problem while ensuring sufficient robustness for the optimisation algorithm. Finally, another advantage of this method is that it is very suitable for parallel computation (Betts and Huffman, 1991).

2.5. DIRECT SOLUTION METHODS AND THE OPTIMALITY PRINCIPLE

An important question which arises after having described the various methods for solving optimal control problems is whether a solution obtained using any of the direct methods would also satisfy the necessary conditions for optimality which have been derived in § 2.2. Some authors have investigated this issue (Enright and Conway, 1990) and the answer to the above question is, in general, yes. As we will describe in a moment, it can be shown that the discrete Lagrange multipliers associated with the solution to an Optimal Control problem obtained by using a direct collocation method are, in fact, a discrete approximation to the solution of the adjoint co-state equations. At this stage, though, it is important to point out that direct solution methods only return *approximate solutions* as a consequence of the problem discretisation. This may lead to some discrepancies with the Optimality Principle, as (Seywald et al., 1996) show in their work. It is obvious from their results, though, that such discrepancies would progressively vanish as one refines the discretisation grid, i.e. by taking shorter time intervals.

Consider a general unconstrained Optimal Control problem. For the sake of simplicity, we shall use for the performance measure the definition of Eq. 2.8:

$$\min_{\mathbf{u}} \quad J = h\left(\mathbf{x}(t_f), t_f\right)$$

$$\textit{subject to}: \quad \dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}, \mathbf{u}, t) \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \qquad \text{Eq. 2.93}$$

$$\textit{for all } t \in [t_0, t_f]$$

By applying the necessary conditions of optimality the adjoint equations that the optimal solution must satisfy are derived. These include the co-state differential equations:

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial \mathbf{a}(\mathbf{x}, \mathbf{u}, t)}{\partial \mathbf{x}}\right)^T \cdot \boldsymbol{\lambda} \qquad \text{Eq. 2.94}$$

with end conditions given by:

$$\boldsymbol{\lambda}(t_f) = \frac{\partial h(\mathbf{x}(t_f))}{\partial \mathbf{x}} \qquad \text{Eq. 2.95}$$

and, finally, the optimality condition:

$$\left(\frac{\partial \mathbf{a}(\mathbf{x},\mathbf{u},t)}{\partial \mathbf{u}}\right)^T \cdot \boldsymbol{\lambda} = \mathbf{0} \qquad \text{Eq. 2.96}$$

Consider now to solve problem 2.93 using a direct collocation method. In order to simplify the next developments, we shall assume the same discretisation grid with N time segments for both control history $\mathbf{u}$ and state trajectory $\mathbf{x}$:

$$\Delta = \left\{ t_0 < t_1 < t_2 < \cdots < t_{N-1} < t_N = t_f \right\} \qquad \text{Eq. 2.97}$$

Furthermore, we shall consider the nodes of the grid evenly spaced, so that the constant length of the time segments reads:

$$\Delta = t_i - t_{i-1} \qquad i = 1,\ldots N \qquad \text{Eq. 2.98}$$

The notation $\mathbf{x}_i$ and $\mathbf{u}_i$ refers to the values of the state and control variables at the i^{th} node respectively, and we shall also use $\mathbf{a}_i$ to refer to the evaluation of the state equations at the same node:

$$\mathbf{a}_i = \mathbf{a}_i(\mathbf{x}_i, \mathbf{u}_i, t_i) \qquad \text{Eq. 2.99}$$

Finally, $\mathbf{x}_N$ and $\mathbf{u}_N$ represent the set of state and control parameters respectively, and the vector of all the independent optimisation variables is:

$$\mathbf{y} = \mathbf{x}_N \cup \mathbf{u}_N \qquad \text{Eq. 2.100}$$

Referring to the trapezoidal discretisation scheme, which was introduced in the previous paragraph, the vector of the defects ξ_i at each node reads:

$$\xi_i = +\mathbf{x}_{i-1} - \mathbf{x}_i + \frac{1}{2} \cdot \Delta \cdot \left[\mathbf{a}_i + \mathbf{a}_{i-1}\right] \qquad i = 1,\ldots N \qquad \text{Eq. 2.101}$$

Problem 2.93 may then be converted into the following Non-Linear Programming problem:

$$\begin{aligned} &\min_{\mathbf{u}_n} \quad (J(\mathbf{y})) \\ &\textit{subject to}: \quad \xi_i(\mathbf{y}) = 0 \qquad i = 1,\ldots N \end{aligned} \qquad \text{Eq. 2.102}$$

Let us now derive the necessary optimality conditions for problem 2.102, i.e. the Kuhn-Tucker conditions. Firstly, we need to form the Lagrangian function:

$$L = J(\mathbf{y}) + \sum_{i=1}^{N} \boldsymbol{\lambda}_i^T \cdot \xi_i \qquad \text{Eq. 2.103}$$

The necessary condition for **y** to be a local constrained minimiser for problem 2.102 is:

$$\nabla_y L = 0 \qquad \text{Eq. 2.104}$$

Consider firstly the state parameters at the interior points of the discretisation grid. Since the objective function J depends exclusively on the final state values, this part of the necessary condition includes only the defects:

$$\frac{\partial L}{\partial \mathbf{x}_k} = \sum_{i=1}^{N} \boldsymbol{\lambda}_i^T \cdot \frac{\partial \boldsymbol{\xi}_i}{\partial \mathbf{x}_k} = \mathbf{0} \qquad k = 1,\ldots,N-1 \qquad \text{Eq. 2.105}$$

But the state parameters $\mathbf{x}_k$ affect only adjacent defects, i.e. ξ_i and ξ_{i+1}, hence Eq. 2.105 reduces to:

$$\boldsymbol{\lambda}_k^T \cdot \frac{\partial \boldsymbol{\xi}_k}{\partial \mathbf{x}_k} + \boldsymbol{\lambda}_{k+1}^T \cdot \frac{\partial \boldsymbol{\xi}_{k+1}}{\partial \mathbf{x}_k} = \mathbf{0} \qquad \text{Eq. 2.106}$$

Using the definition for the defects of Eq. 2.101 in Eq. 2.106 yields:

$$\boldsymbol{\lambda}_k^T \cdot \left(-\mathbf{I} + \frac{\Delta}{2} \cdot \frac{\partial \mathbf{a}_k}{\partial \mathbf{x}_k} \right) + \boldsymbol{\lambda}_{k+1}^T \cdot \left(\mathbf{I} + \frac{\Delta}{2} \cdot \frac{\partial \mathbf{a}_k}{\partial \mathbf{x}_k} \right) = \mathbf{0} \qquad \text{Eq. 2.107}$$

where **I** is the identity matrix. Rearranging the terms in Eq. 2.107 we finally obtain:

$$\boldsymbol{\lambda}_k^T - \boldsymbol{\lambda}_{k+1}^T - \frac{\Delta}{2} \cdot \left(\boldsymbol{\lambda}_k^T + \boldsymbol{\lambda}_{k+1}^T \right) \cdot \frac{\partial \mathbf{a}_k}{\partial \mathbf{x}_k} = \mathbf{0} \qquad \text{Eq. 2.108}$$

Eq. 2.108 is clearly a discrete version of the adjoint equations 2.94. In fact, we may approximate Eq. 2.94 over a time segment as follows:

$$\int_{t_k}^{t_{k+1}} \dot{\boldsymbol{\lambda}}\, dt = -\int_{t_k}^{t_{k+1}} \left(\frac{\partial \mathbf{a}(\mathbf{x},\mathbf{u},t)}{\partial \mathbf{x}} \right)^T \cdot \boldsymbol{\lambda}\, dt \qquad \text{Eq. 2.109}$$

Assuming that the Jacobian in the right hand term of Eq. 2.109 is constant over the time interval h we may write:

$$\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k \cong -\left(\frac{\partial \mathbf{a}_k}{\partial \mathbf{x}_k} \right)^T \cdot \int_{t_k}^{t_{k+1}} \boldsymbol{\lambda}\, dt \cong -\left(\frac{\partial \mathbf{a}_k}{\partial \mathbf{x}_k} \right)^T \cdot \frac{\Delta}{2} \cdot \left(\boldsymbol{\lambda}_{k+1} + \boldsymbol{\lambda}_k \right) \qquad \text{Eq. 2.110}$$

Consider now the end point of the time interval. The necessary condition now reads:

$$\frac{\partial L}{\partial \mathbf{x}_N} = \frac{\partial h(\mathbf{x}_N)}{\partial \mathbf{x}_N} + \boldsymbol{\lambda}_N^T \cdot \frac{\partial \boldsymbol{\xi}_N}{\partial \mathbf{x}_N} = \mathbf{0} \qquad \text{Eq. 2.111}$$

Proceeding as above, differentiating the defects with respect to $\mathbf{x}_N$ provides the terminal boundary conditions for Eq. 2.108:

$$\boldsymbol{\lambda}_N^T - \boldsymbol{\lambda}_N^T \cdot \frac{\Delta}{2} \cdot \frac{\partial \mathbf{a}_N}{\partial \mathbf{x}_N} = \frac{\partial h(\mathbf{x}_N)}{\partial \mathbf{x}_N} \qquad \text{Eq. 2.112}$$

which corresponds to the condition 2.95. Let us finally consider the partial derivative of the Lagrangian with respect to the control variables at the interior nodes:

$$\frac{\partial L}{\partial \mathbf{u}_k} = \sum_{i=1}^{N} \boldsymbol{\lambda}_i^T \cdot \frac{\partial \boldsymbol{\xi}_i}{\partial \mathbf{u}_k} = \mathbf{0} \qquad k = 1,\ldots,N-1 \qquad \text{Eq. 2.113}$$

Since the control parameters $\mathbf{u}_k$ only affect adjacent defects, Eq. 2.113 simplifies as follows:

$$\boldsymbol{\lambda}_k^T \cdot \frac{\partial \boldsymbol{\xi}_k}{\partial \mathbf{u}_k} + \boldsymbol{\lambda}_{k+1}^T \cdot \frac{\partial \boldsymbol{\xi}_{k+1}}{\partial \mathbf{u}_k} = \mathbf{0} \qquad \text{Eq. 2.114}$$

Differentiating the defects and substituting the result in Eq. 2.114 yields:

$$\boldsymbol{\lambda}_k^T \cdot \left(\frac{\Delta}{2} \cdot \frac{\partial \mathbf{a}_k}{\partial \mathbf{u}_k} \right) + \boldsymbol{\lambda}_{k+1}^T \cdot \left(\frac{\Delta}{2} \cdot \frac{\partial \mathbf{a}_k}{\partial \mathbf{u}_k} \right) = \mathbf{0} \qquad \text{Eq. 2.115}$$

Rearranging the terms we finally obtain:

$$\left(\boldsymbol{\lambda}_k^T + \boldsymbol{\lambda}_{k+1}^T \right) \cdot \frac{\Delta}{2} \cdot \frac{\partial \mathbf{a}_k}{\partial \mathbf{u}_k} = \mathbf{0} \qquad \text{Eq. 2.116}$$

Eq. 2.116 is a discretised version of the optimality condition 2.96. Therefore we may conclude that the solution to the discretised problem also satisfies the Optimality Principle. It is important to observe, though, that the discretised adjoint equations 2.108 and 2.116 provide lower accuracy than that ensured by the discretised state differential equation.

2.6. CONCLUSIONS

A review of Optimal Control theory has been presented in this section. Various methods for solving Optimal Control problems have been described. The methods have been identified in two distinct classes, named direct and indirect methods. The practical implications involved in following the two different approaches have been highlighted. The range of possibilities is rather wide, including also strategies which involve the

combination of two different methods in order to achieve a particular goal. For example, some authors suggest the use of a direct method in combination with an indirect method (Stryk et al., 1993). According to the theory described above, the direct method allows to obtain an initial estimate of the adjoint variables. This is then used as initial guess for the indirect method in order to achieve greater accuracy in the solution.

Although the choice of the ideal solution strategy for the minimum lap time optimisation problem will be discussed later on, direct methods are favoured here. First of all, the aim is to develop a procedure which does not impose any constraint to the modelling of the vehicle, as look-up tables and interpolation techniques are conveniently used to model various vehicle components. From this point of view, the fact that no symbolic manipulation of the problem equations is required when applying a direct solution method is seen as an advantage. Another important aspect of direct method is that they naturally suit problems with discontinuous controls. As we shall see, this is a feature of the car longitudinal control when switching from full driving into applying the brakes. Finally, a good deal of technology has been recently developed in the field of numerical optimisation. Sparse SQP algorithms allow to solve large scale problems involving thousands of independent optimisation parameters, and Automatic Differentiation techniques allow to compute derivatives with greater efficiency and accuracy compared to more traditional numerical differentiation techniques.

The downside of direct solution methods is their approximative nature. This issue needs to be investigated by observing the behaviour of the solution for different discretisation strategies. However, also indirect methods require some sort of discretisation when a digital computer is used to solve the non-linear, two-point boundary value problem involved, hence involving similar problems.

Chapter 3.

Racing Line Reconstruction from Telemetry Data

The reconstruction of the racing line is an important step in the analysis of on-board measured vehicle data. For example, a computer model of the race car may be used to follow the reconstructed path and the theoretical results may be compared with the measurements directly. The analysis software which is commercially available is normally capable to automatically generate the racing line from the data acquired[1]. However, to date no literature exists on the subject.

In this section the problem of the reconstruction of the racing line from on-board vehicle measurements is examined and a method for doing it is presented. The basic approach uses a simple kinematic model of the car to estimate the curvature of its trajectory using the forward velocity and the lateral acceleration. Then, the co-ordinates of the trajectory are evaluated. This involves the double integration of the curvature with respect to the travelled distance.

Because of the various sources of imperfection, both in the model and in the data, the trajectory obtained by applying this basic method will not, in general, be a closed loop. As a consequence of the double integration involved, the error which is inevitably distributed along the curvature is enormously amplified and this results in a large gap between the start and the end of the trajectory. Particularly, errors in the measurements recorded in the early stage of the lap will have a much greater influence on the final gap since they affect all of the subsequent trajectory. As the gap may be as large as hundreds of meters, in many cases the trajectory which is obtained may bear little resemblance to the circuit.

Errors in the measured data essentially include randomly distributed noise, the natural off-set of the measuring device and possible errors in the calibration of the instrumentation. As far as the simple kinematic model is concerned, it does not take into account the longitudinal slip of the front wheel from which the forward vehicle velocity is taken. Also, the tyre growth in diameter at high speed and the vehicle lateral drift are not included in the model. All these causes contribute to the error in the racing line reconstruction. A strategy to compensate their effects is essential.

In the procedure which will be described here the measured data are firstly processed in order to eliminate the high-frequency noise which is normally present. This is done in two phases, the first using digital filtering and the second using spline interpolation techniques. The elimination of the noise, though, is not in general

[1] For further information, see: PI-Research on www.pi-group.com; Magneti Marelli Competition Systems Ltd. on www.mmcompsys.com, or Racecar Engineering, Vol. 10N5, June 2000, pp 55-60.

sufficient to reconstruct the racing line with acceptable accuracy. Two strategies which allow to close up the trajectory at the end have then been devised. One is purely empirical and simply involves "stretching" the path co-ordinates till the gap between the start and the end of the trajectory coincide. The second method is based on the physical nature of the typical errors which are thought to be involved in the measurements. Both algorithms have been implemented and tested and the results are presented at the end of this section.

3.1. THE BASIC KINEMATIC MODEL

Let V_{vx} be the vehicle forward velocity and a_{vy} be the vehicle lateral acceleration. These quantities are measured on-board a real race car as functions of the time elapsed from the start of a lap of the circuit. Neglecting the vehicle lateral drift, the local curvature of the vehicle path may be approximated by the lateral acceleration divided by the square of the forward velocity, as described by:

$$k_t(t) = \frac{1}{r_t(t)} \cong \frac{a_{vy}(t)}{V_{vx}^2(t)} \qquad \text{Eq. 3.1}$$

Next, consider a reference axis system fixed in space and located at the start of the vehicle trajectory. The trajectory is described by its co-ordinates x_t and y_t, the tangent angle ψ_t and the local turning radius r_t (or the curvature k_t) as functions of the path length co-ordinate s (figure 3.1).

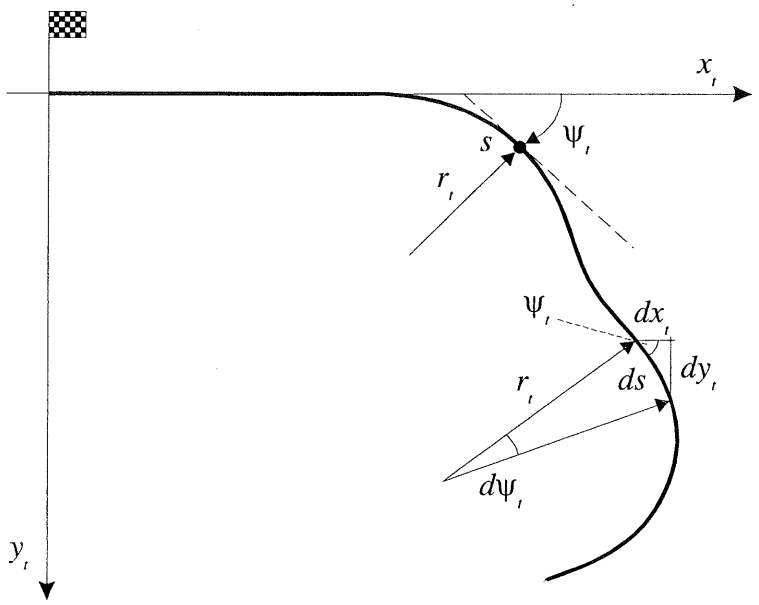


Figure 3.1 Racing line description.

As can be derived from figure 3.1, these parameters are not independent. Rather, the following differential relationships hold:

$$ds = r_t \cdot d\psi_t \Rightarrow d\psi_t = \frac{1}{r_t} \cdot ds = k_t \cdot ds \qquad \text{Eq. 3.2}$$

$$dx_t = \cos(\psi_t) \cdot ds \qquad \text{Eq. 3.3}$$

$$dy_t = \sin(\psi_t) \cdot ds \qquad \text{Eq. 3.4}$$

The integration of Eqs. 3.2, 3.3 and 3.4 yields the path tangent angle and the path co-ordinates. However, since Eq. 3.1 returns the curvature as a function of time, it is firstly necessary to change the independent variable using the relationship:

$$ds = V_{vx} \cdot dt \qquad \text{Eq. 3.5}$$

Then, replacing *ds* in Eqs. 3.2 to 3.4 and integrating over time, one may write:

$$\psi_t(t) = \int_0^t k_t(t) \cdot V_{vx}(t) \cdot dt \qquad \text{Eq. 3.6}$$

$$x_t(t) = \int_0^t \cos(\psi_t(t)) \cdot V_{vx}(t) \cdot dt \qquad \text{Eq. 3.7}$$

$$y_t(t) = \int_0^t \sin(\psi_t(t)) \cdot V_{vx}(t) \cdot dt \qquad \text{Eq. 3.8}$$

Finally, the integration of Eq. 3.5 yields the travelled distance as a function of the elapsed time:

$$s(t) = \int_0^t V_{vx}(t) \cdot dt \qquad \text{Eq. 3.9}$$

This can be used to express the trajectory parameters as function of the path co-ordinate *s*, which is more convenient for the further use of the map of the racing line.

3.2. DATA FILTERING

Figure 3.2 shows a typical recording of the vehicle longitudinal velocity and lateral acceleration for one lap of the circuit in Barcelona, Spain. In order to eliminate the noise in the signal a digital lowpass filter was designed with the aid of the MATLAB® Signal Processing Toolbox (Anon. 1996). The convention for the filter design functions is to operate with the *normalised frequency*, i.e. the frequency divided by half the sampling rate. The normalised frequency varies within the interval [0, 1], where the unit frequency is the *Nyquist frequency*, defined as half the sampling rate.

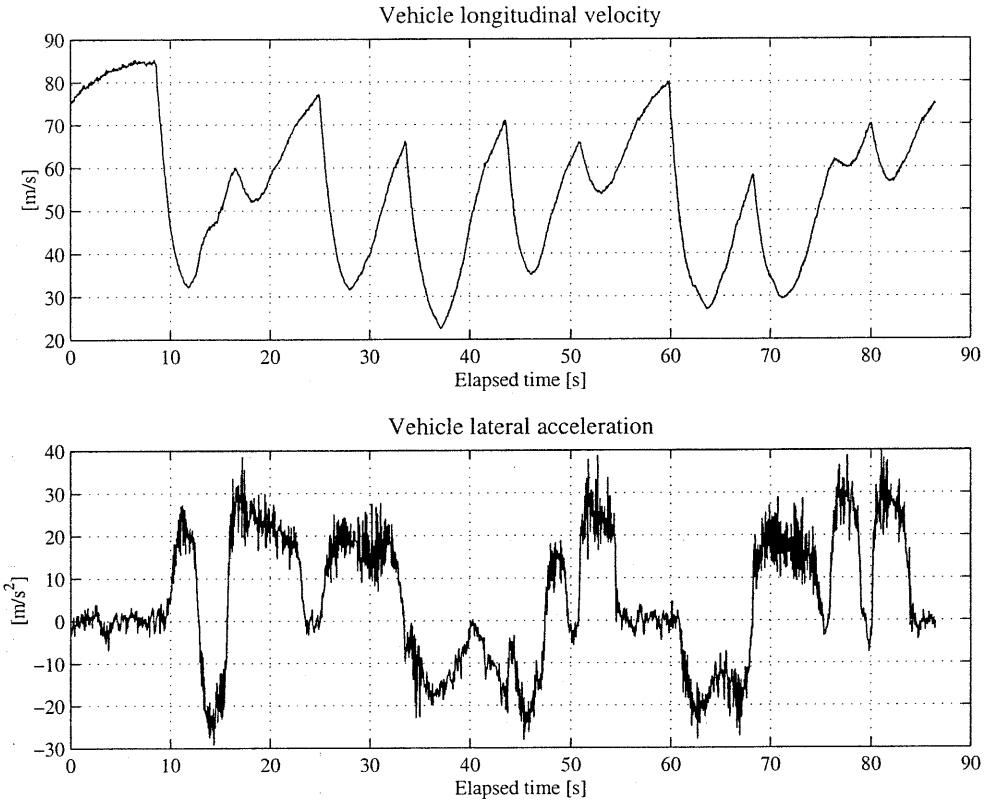


Figure 3.2 Vehicle longitudinal velocity and lateral acceleration (Barcelona).

Various functions are available for filter design which approximate an ideal filter in different ways. For the purpose of the present work, the *Chebyshev type II* filter was chosen. The important features of this filter are that its passband response is very flat and the transition between the passband and the stopband is relatively sharp compared to other filters, e.g. the *Butterworth* filter. Figure 3.3 shows the transfer function for a 13^{th} order Chebyshev type II filter with the cutoff frequency W_n equal to 0.2 and a required stopband attenuation of 80 dB. Note that this type of digital filter introduces a phase distortion, but this can easily be avoided by processing the data with the Matlab function `filtfilt` (Anon., 1996).

The main problem which arises when one applies the digital filter to the measured data lies in the nature of the signal. As is visible in figure 3.2, certain parts of the signal vary much faster than others, e.g. the sharp variation of the velocity when the vehicle brakes or the rapid build up of the lateral acceleration as the vehicle turns. Inevitably, if one reduces the filter cutoff frequency to better smooth the slowly-varying parts of the signals, other parts of the signals which feature more rapid variations will be badly affected. Figure 3.4 shows a detail of the first forty seconds of the original vehicle longitudinal velocity depicted in figure 3.2 after the digital filter has been applied with four different values for the cutoff frequency (for this case the data sampling rate was 20 Hz). As the signal becomes smoother for the lower values of the cutoff frequency, oscillations are introduced in the original data, mostly in correspondence to the areas where the vehicle shifts from driving to braking.

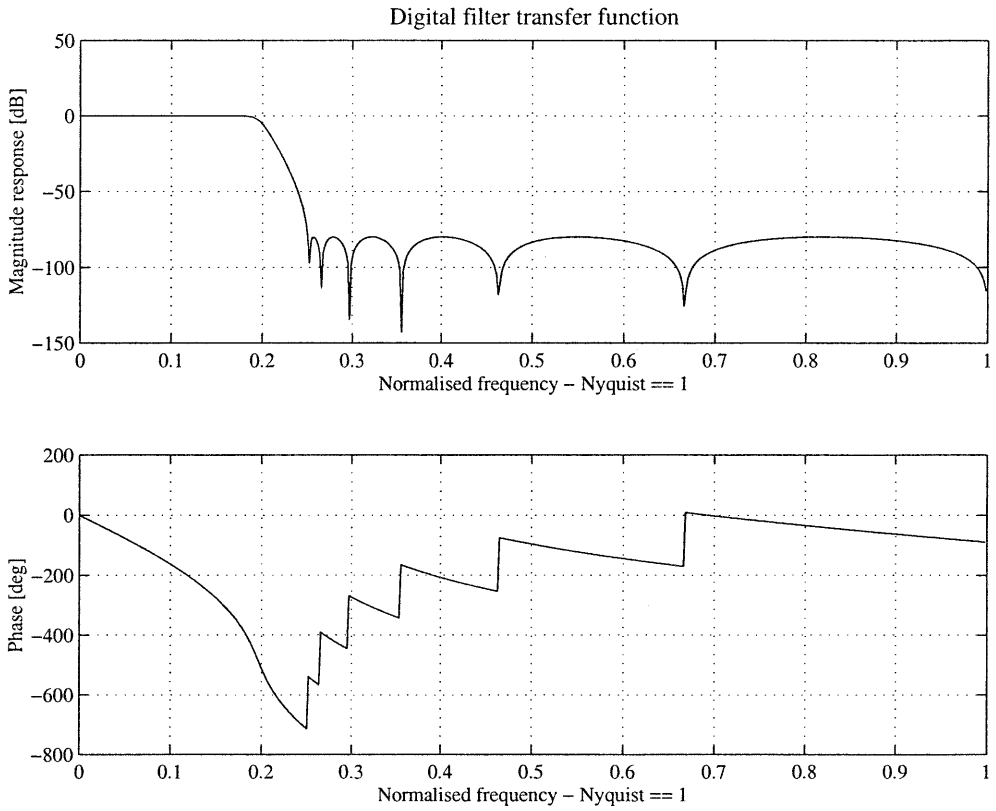


Figure 3.3 Chebyshev type II digital filter transfer function for $W_n = 0.2$.

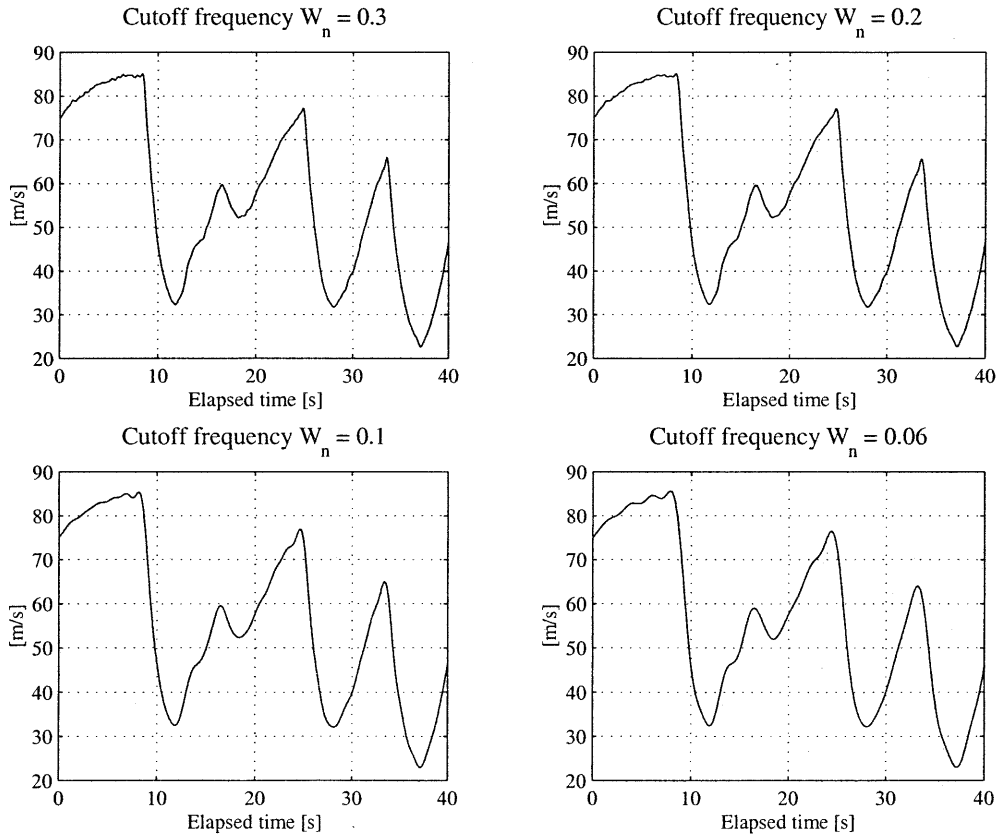


Figure 3.4 Data filtering trials at different cutoff frequencies.

In order to avoid this problem, it was chosen to use the digital filter with a relatively high cutoff frequency in combination with spline interpolation techniques. Cubic splines allow to interpolate discrete points with a smooth line while avoiding undesired oscillations between the points, as would be the case when using other techniques, e.g. polynomial interpolation. The properties of the cubic splines may be used advantageously here to further smooth the data. After the digital filtering, the data are interpolated by selecting a fixed grid of evenly spaced points. In order to achieve the desired effect, the selection of the density of the grid of points for the interpolation is crucial. If the density is too high, the spline interpolating line will still match the irregularities in the data and will not yield any improvement. Conversely, if the density is too low, the interpolation will be poor where the data vary rapidly. The appropriate grid for the spline interpolation was devised after some trials and it finally consists of evenly spaced points 0.5 [s] far apart from each other. The final results are shown in figures 3.5 and 3.6. The first figure refers to the data filtered using W_n equal to 0.2, and the second figure refers to the final data after the spline interpolation was performed.

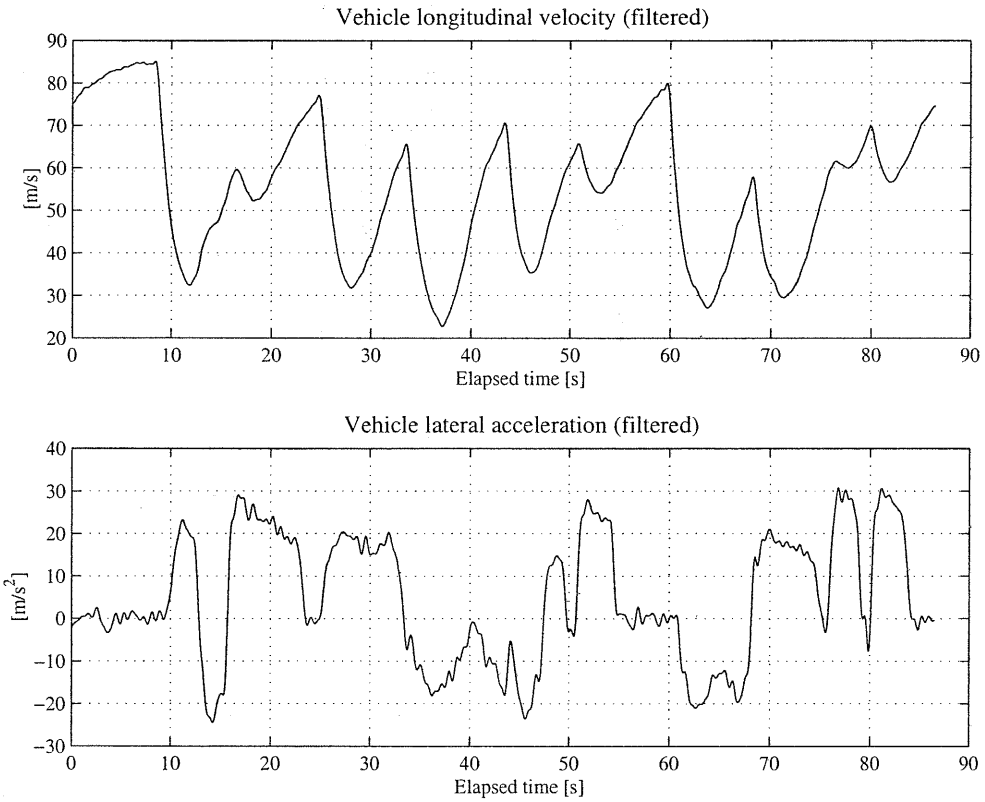


Figure 3.5 Vehicle longitudinal velocity and lateral acceleration, filtered.

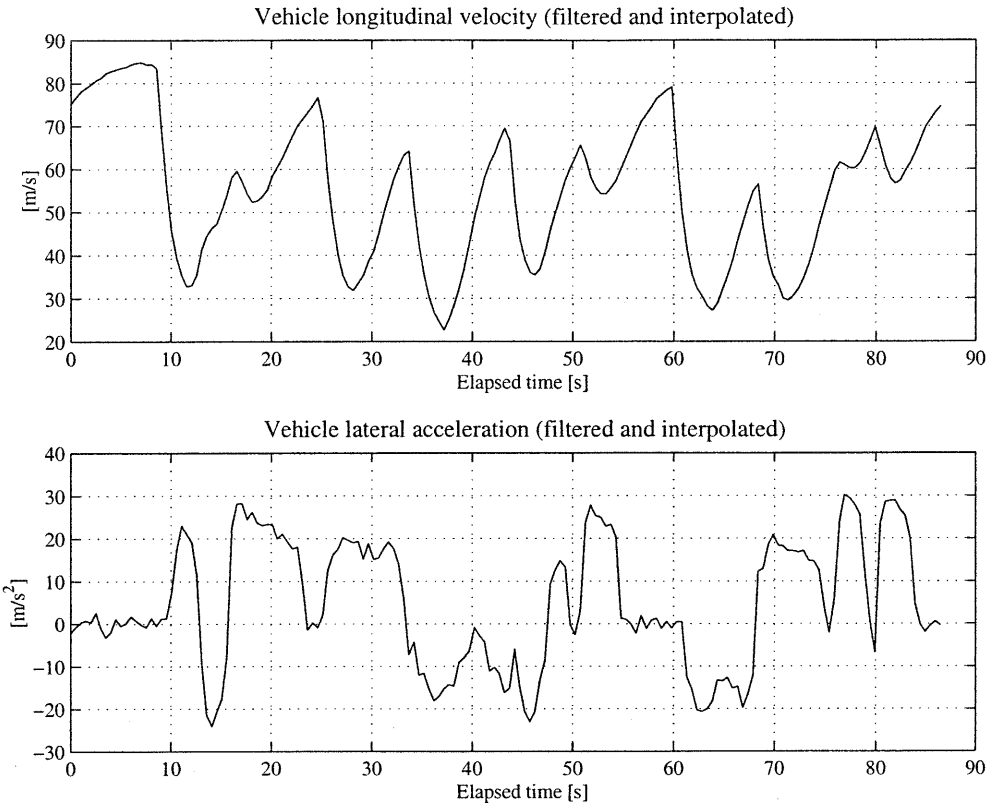


Figure 3.6 Vehicle longitudinal velocity and lateral acceleration, filtered and interpolated.

3.3. THE "STRETCHING" ALGORITHM

If the vehicle longitudinal velocity and lateral acceleration for one lap of the circuit in Barcelona, obtained after the signal processing, are used to reconstruct the vehicle trajectory according to the basic kinematic model described in §3.1, the result will be that shown in figure 3.7. Formula One enthusiasts should be able to recognise the shape of the Catalunya circuit. However, the data still contain imperfections which, after the double integration process, determine a substantial gap between the ends of the trajectory.

In the approach presented here for closing such gap between the start and the end of the reconstructed racing line, we shall not try to investigate what those imperfections might be. Instead, we will look at their overall effect on the results and apply empirical corrections. First of all, by observing figure 3.7 the angular gap between the ends of the trajectory may be interpreted as if the trajectory had been bent slightly towards the right hand side of the direction of travel for its entire length. This, ideally, corresponds to the case of a small positive constant added to the curvature of the racing line. Hence, adding a small constant with opposite sign should eliminate the angular gap and bring the ends of the reconstructed trajectory much closer to each other. The procedure which then arises is based on enforcing the two ends of the trajectory to have the same tangent angle.

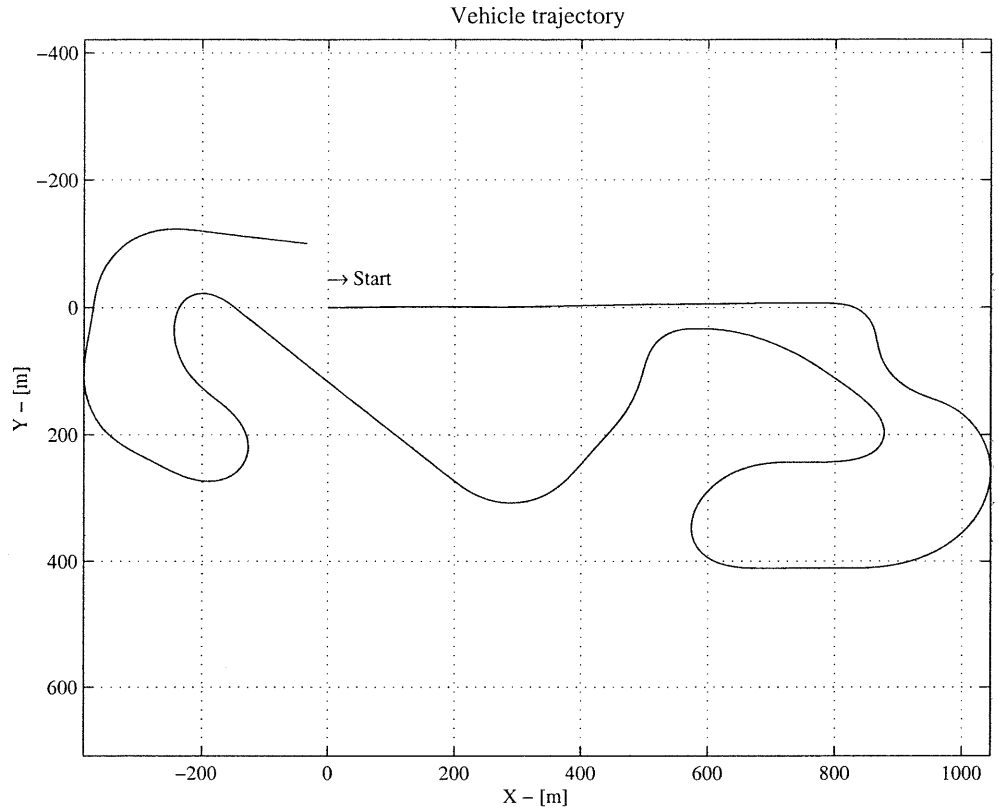


Figure 3.7 Racing line reconstructed using the basic kinematic model.

Firstly, the integration of Eq. 3.5 over the duration of the lap yields the length of the trajectory:

$$S = \int_0^T V_{vx}\, dt \qquad \text{Eq. 3.10}$$

Then, the path curvature may be estimated using Eq. 3.1 with the filtered and interpolated experimental data, and Eq. 3.6 may be solved to obtain the first estimate of the tangent angle of the racing line. Since the tangent angle at the start of the trajectory is always equal to zero, we may define the final path tangent error as follows:

$$E_{\psi} = \psi_t(T) - 2\pi \qquad \text{Eq. 3.11}$$

Eq. 3.11 holds for all the circuits without overhead crossing, where after one lap the vehicle will have turned a full 360 degrees with respect to the reference axis system fixed in space. Instead, for the case of circuits with overhead crossing, e.g. Suzuka, Eq. 3.11 simply reduces to:

$$E_{\psi} = \psi_t(T) \qquad \text{Eq. 3.12}$$

In order to zero the final path tangent error, the necessary correction for the curvature reads:

$$\Delta_{\psi} = -\frac{E_{\psi}}{S} \qquad \text{Eq. 3.13}$$

Such correction is equivalent to the overall path tangent error being evenly distributed along the entire trajectory, but with opposite sign. Hence, the evaluation of Eq. 3.6 after applying the result returned by Eq. 3.13 yields the corrected path tangent $\overline{\psi}_t(t)$ which satisfies the condition $E_{\psi} = 0$:

$$\overline{\psi}_t(t) = \int_0^t \left(k_t(t) + \Delta_{\psi}\right) \cdot V_{vx}(t) \cdot dt \qquad \text{Eq. 3.14}$$

Finally, Eqs. 3.7 and 3.8 may be solved using the above result, and the outcome will be that shown in figure 3.8. Although the result is improved and the angular gap has vanished, the two ends of the trajectory are still several metres far apart from each other. The next task will therefore be to close the residual gap while maintaining the continuity of the tangent angle across the start-finish line.

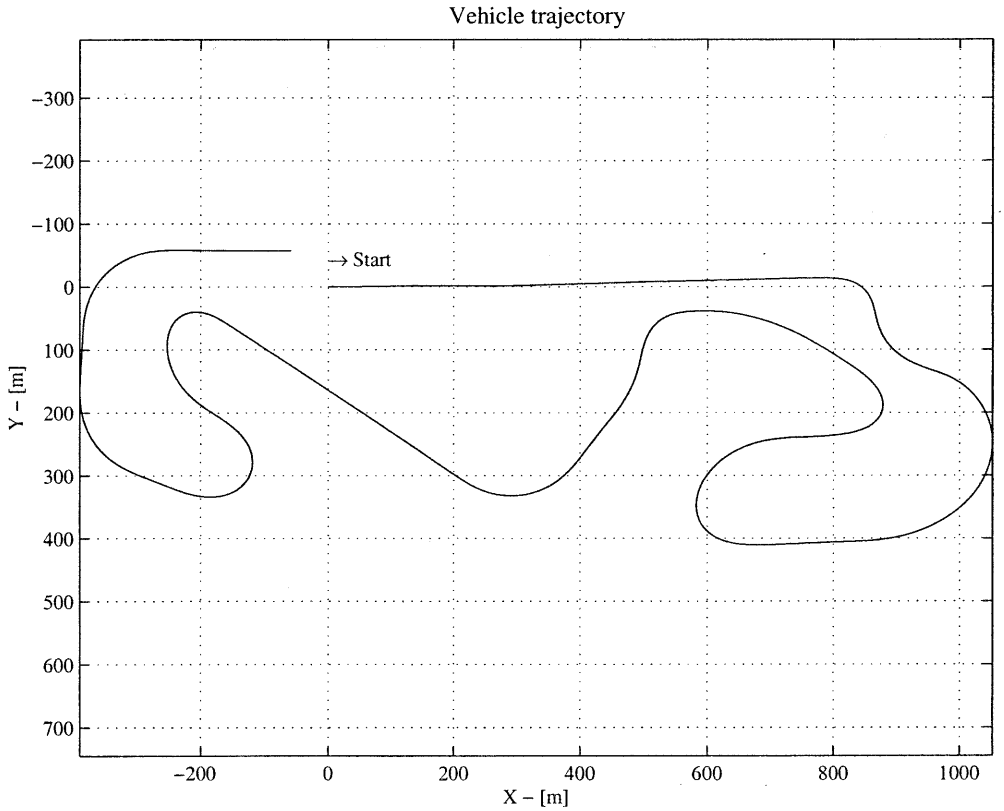


Figure 3.8 Racing line reconstruction with path tangent error compensation.

In order to do so, we shall proceed as follows. Let us consider that the correction of the final path tangent error has been successfully completed and $E_{\psi} = 0$. Next, Eqs. 3.7 and 3.8 are solved to evaluate the first estimate of the path co-ordinates. Since the final co-ordinates of the trajectory must coincide with the origin of the reference axis system, the final x and y co-ordinate errors read:

$$E_x = x_t(T); \qquad E_y = y_t(T) \qquad \text{Eq. 3.15}$$

Then, the necessary corrections to zero the final trajectory gap will be:

$$\Delta_x = -\frac{E_x}{S}; \qquad \Delta_y = -\frac{E_y}{S} \qquad \text{Eq. 3.16}$$

Finally, the path co-ordinates may be re-computed using Eqs. 3.7 and 3.8 after adding the correcting constants returned by Eq. 3.16:

$$\bar{x}_t(t) = \int_0^t [\cos(\bar{\psi}_t(t)) + \Delta_x] \cdot V_{vx}(t) \cdot dt \qquad \text{Eq. 3.17}$$

$$\bar{y}_t(t) = \int_0^t [\sin(\bar{\psi}_t(t)) + \Delta_y] \cdot V_{vx}(t) \cdot dt \qquad \text{Eq. 3.18}$$

This procedure may be interpreted as "stretching" the path derivatives stated by Eqs. 3.3 and 3.4, hence the name of the algorithm. Although the precision depends on the accuracy with which the various integrations are performed, one single iteration reduces the gap to a negligible value. Figure 3.9 shows the result so far.

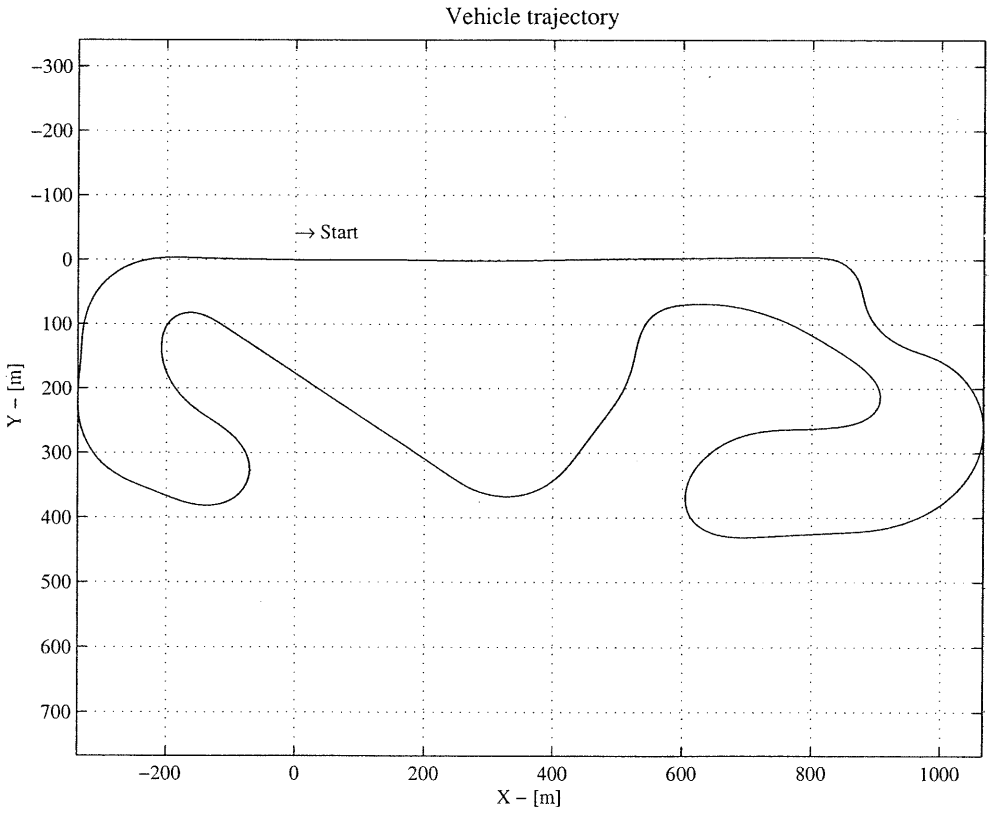


Figure 3.9 Racing line reconstructed using the "stretching" algorithm.

As a consequence of the correction applied to the path co-ordinates in Eqs. 3.17 and 3.18, the actual path tangent will be different from that obtained from Eq. 3.14. Hence, it will be necessary re-compute it by using the following relationship, derived from figure 3.1:

$$\overline{\overline{\psi}}_t = \arctan\left(\frac{d\overline{y}_t}{d\overline{x}_t}\right) \qquad \text{Eq. 3.19}$$

Note, though, that the tangent angle will still be the same for both the start and the end of the trajectory. In fact, assuming that the original path tangent $\overline{\psi}_t(t)$ from Eq. 3.14 satisfies $E_\psi = 0$, the following relationships hold:

$$\begin{aligned} \sin(\overline{\psi}(0)) &= \sin(\overline{\psi}(T)) \\ \cos(\overline{\psi}(0)) &= \cos(\overline{\psi}(T)) \end{aligned} \qquad \text{Eq. 3.20}$$

Thus, adding the corrections yielded by Eq. 3.16 to the path derivative equations 3.3 and 3.4, by virtue of Eq. 3.20 we may write:

$$\begin{aligned} \left(\frac{d\overline{x}_t}{ds}\right)_{t=0} &= \cos(\overline{\psi}_t(0)) + \Delta_x = \left(\frac{d\overline{x}_t}{ds}\right)_{t=T} = \cos(\overline{\psi}_t(T)) + \Delta_x \\ \left(\frac{d\overline{y}_t}{ds}\right)_{t=0} &= \sin(\overline{\psi}_t(0)) + \Delta_y = \left(\frac{d\overline{y}_t}{ds}\right)_{t=T} = \sin(\overline{\psi}_t(T)) + \Delta_y \end{aligned} \qquad \text{Eq. 3.21}$$

This is sufficient to prove the continuity of the path tangent angle across the start-finish line of the reconstructed vehicle trajectory.

3.4. ERROR COMPENSATION ALGORITHM

A second strategy for reconstructing the racing line without any gap between its ends involves using three different physically based corrections, which are applied to the curvature estimated using Eq. 3.1, in order to zero the three errors E_ψ, E_x and E_y which were defined in Eqs. 3.12 and 3.15. These corrections are meant to compensate the errors which are thought to affect the measurements even after processing the measured data as was described in §3.2. They consist of a constant off-set, a linear, time dependent off-set and a proportional correction. The expression for the curvature correction then reads:

$$\overline{k}_t(t) = p_1 + p_2 \cdot t + p_3 \cdot k_t(t) \qquad \text{Eq. 3.22}$$

where $k_t(t)$ is obtained from Eq. 3.1. The actual curvature is then thought of as the sum of that obtained directly from the measurements and the corrections. The constant off-set is meant to compensate the natural off-set of the measuring device. For example, if a positive constant were added to the lateral acceleration, the effect on the path curvature would be in such a way that the right hand corners would have a tighter radius, while

the left hand corners would be more open. The linear, time dependent off-set is applied in order to balance the effect of the drift induced by the integration as a consequence of the randomly distributed error. Finally, the proportional correction is intended to account for possible errors in the calibration of the instrumentation.

Using the curvature returned by Eq. 3.22, the path tangent angle and the path co-ordinates may be evaluated using Eqs. 3.6, 3.7 and 3.8. Then, the final gap is evaluated in terms of the three errors E_ψ, E_x and E_y. The task is then to find the compensation parameters p_1, p_2 and p_3 which allow to zero the end gap in the trajectory. As the three errors E_ψ, E_x and E_y may be expressed directly as functions of the compensation parameters, the task is set-up as the following multi-dimensional root finding problem:

$$\begin{aligned} &\textit{find} && \mathbf{p} = \{p_1 \quad p_2 \quad p_3\} \\ &\textit{such that} && E_\psi(\mathbf{p}) = E_x(\mathbf{p}) = E_y(\mathbf{p}) = 0 \end{aligned} \qquad \text{Eq. 3.23}$$

Problem 3.23 is solved using a least squares method.

One advantage of the "error compensation" algorithm compared to the "stretching" algorithm is that it yields the path curvature, the path tangent and the path co-ordinates at the same time. This avoids the need for the final refinement of the results which was dealt with in Eq. 3.19. One disadvantage is that the iterative solver requires a careful scaling of the problem parameters. As the sensitivity of the three errors E_ψ, E_x and E_y with respect to the three compensation parameters may vary a great deal, the Jacobian of the problem may be very badly scaled and the solver may fail to converge or may converge to meaningless solutions.

3.5. RESULTS

Two sets of data, each including the vehicle longitudinal velocity and lateral acceleration recorded for three laps during the warm-up of the 1999 Spanish and Japanese Grand Prix, have been used to reconstruct the racing lines using both the algorithms described above. The results obtained using the "stretching" algorithm are presented in figures 3.10 and 3.11, while the results obtained using the "error compensation" algorithm are shown in figures 3.12 and 3.13. In each case the three racing lines are compared with the map of the centre line of the circuit. Finally, Table 3.1 reports the lengths of each of the reconstructed racing lines together with the reference circuit length.

RACING LINE LENGTHS				
	BARCELONA		SUZUKA	
Reference circuit length	4728 m		5864 m	
Algorithms	Stretching	Error comp.	Stretching	Error comp.
Lap 1	4727 m	4728 m	5830 m	5860 m
Lap 2	4726 m	4727 m	5848 m	5861 m
Lap 3	4727 m	4729 m	5845 m	5860 m

Table 3.1 Lengths of the reconstructed racing lines.

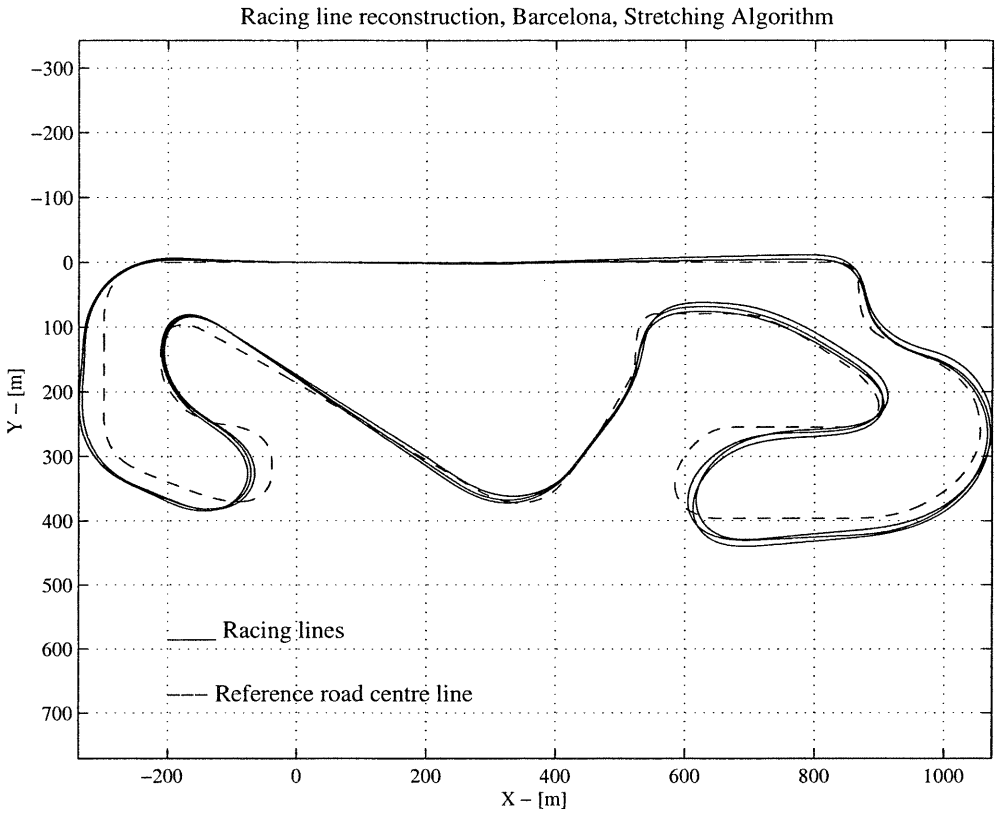


Figure 3.10 Racing lines obtained with the stretching algorithm, Barcelona.

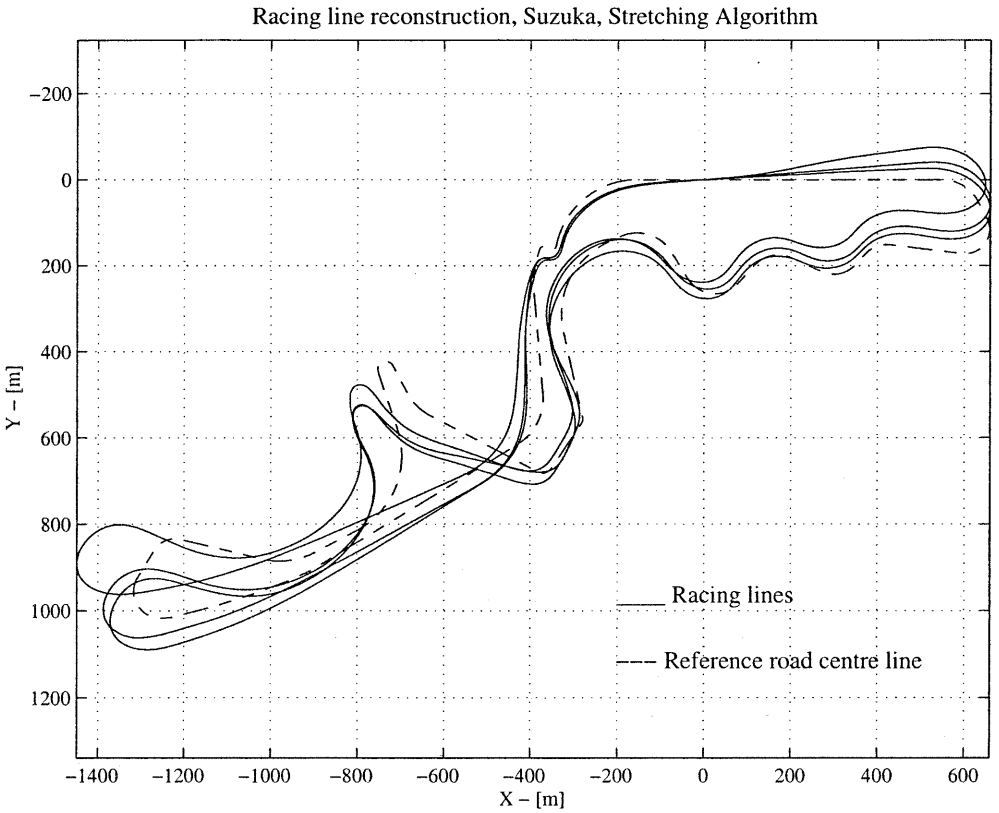


Figure 3.11 Racing lines obtained with the stretching algorithm, Suzuka.

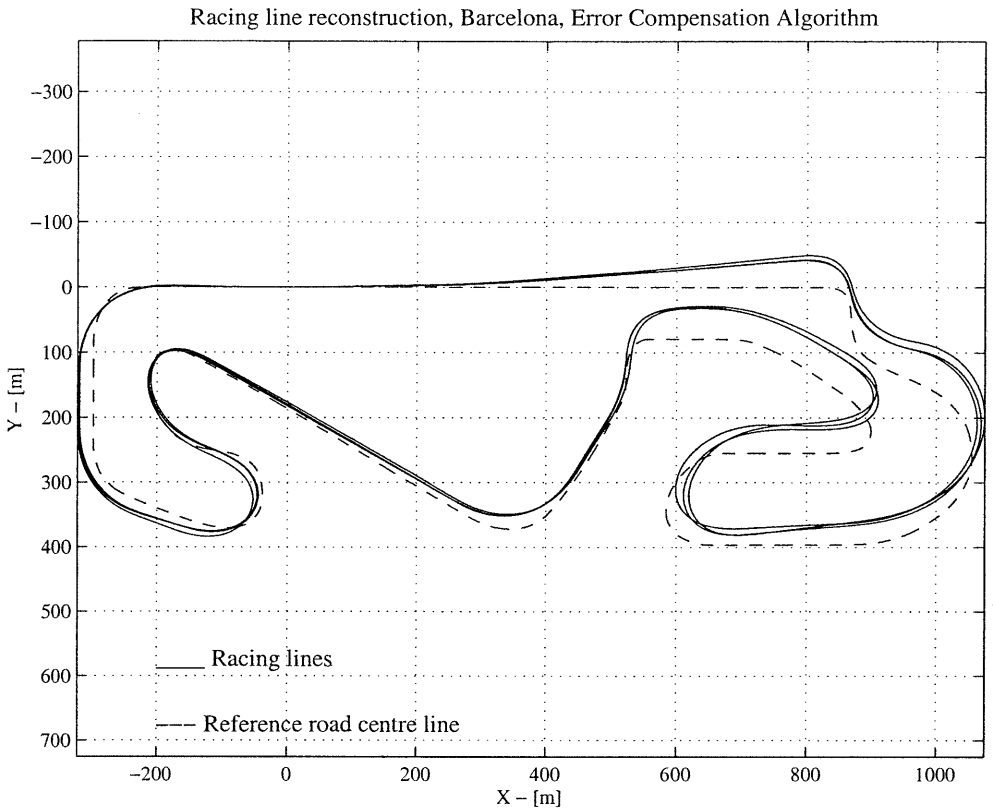


Figure 3.12 Racing lines obtained with the error compensation algorithm, Barcelona.

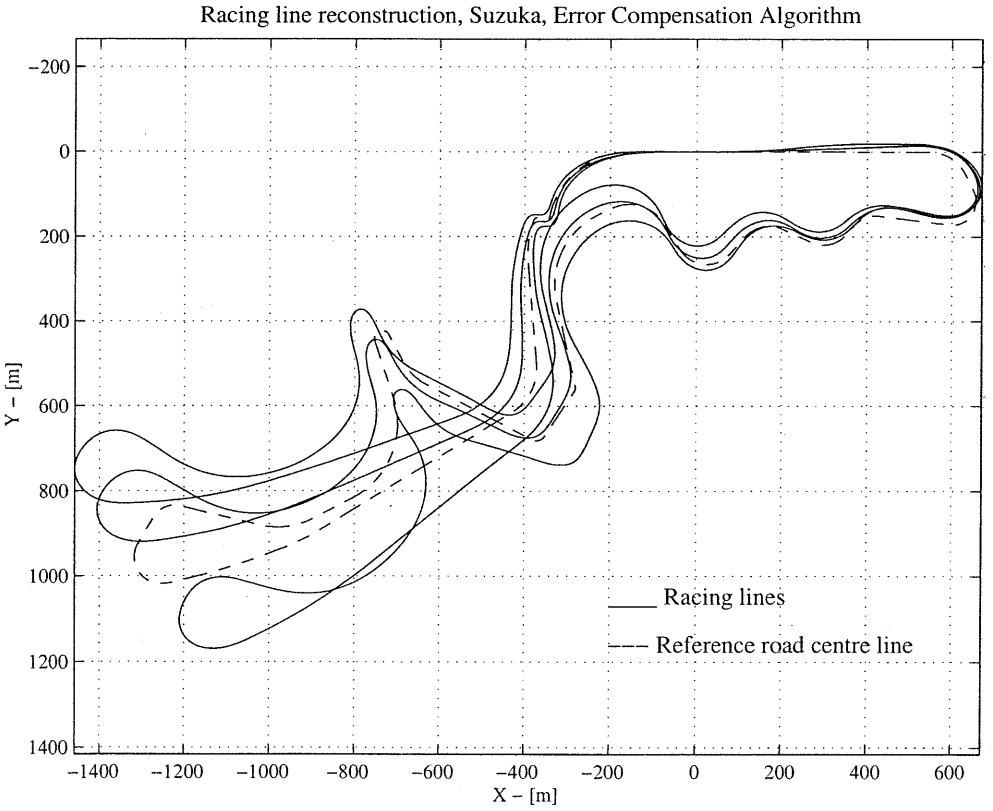


Figure 3.13 Racing lines obtained with the error compensation algorithm, Suzuka.

3.6. CONCLUSIONS

In this section the use of on-board measured vehicle data for the reconstruction of the racing line has been investigated. Two algorithms to perform this task have been described and some results have been presented in comparison with maps of the circuit where the laps were recorded. Although both algorithms are capable to return reasonable estimates of the racing lines, the "stretching" algorithm usually yields better and more consistent results. Even though there were no distinguishable features in the set of data from the laps of the Barcelona circuit compared to the set of data from the laps of Suzuka, the former yielded much better and more consistent results. This shows how the reconstruction procedure is extremely sensitive to small imperfections in the estimated curvature as a consequence of the double integration involved.

Particular emphasis was put on processing the measured data in order to remove the random noise which normally accompanies the recorded signals. A strategy based on a combination of digital filtering and spline interpolation techniques has been devised in order to accurately smooth the data without altering their fundamental features. The values for the parameters involved which have been suggested in §3.2 have been used for various cases with satisfactory results and are not meant to be adjusted by the user of the software. It is important to point out, though, that the random noise in the measured data was not found to be as influential on the racing line reconstruction procedure as other facts, like, for example, a constant off-set in the measuring device. This is because the double integration itself has a smoothing effect so that the resulting trajectory will not be noisy. However, obtaining very smooth signals for the vehicle longitudinal velocity and for the curvature of the racing line is very important in view of using these data for solving path following tasks by employing a driver model, as we shall see later on.

In the "error compensation" algorithm three parameters were needed in order to reduce to zero three error functions representing the gap between the ends of the trajectory. Such parameters were made representative of the three main errors which are thought to affect the measurements, i.e. the instrument off-set, the integration drift resulting from random noise and the calibration error, and a curvature correction was defined with Eq. 3.22. This method always succeeds in closing up the gap, but often the trajectory obtained is distorted compared with the equivalent solutions obtained using the "stretching" algorithm, as is clearly visible comparing figure 3.12 with figure 3.10. The reason for this is that probably Eq. 3.22 is not truly representative of the errors in the reconstruction procedure and it also constrains the error compensation to take a certain specified form. The consequence is that, although a solution is found anyway, the trajectory is distorted in order to accommodate the imperfections which are not accounted for.

In this work we mainly focused on the errors which affect the measurements. However, the simple kinematic model which constitutes the base of the method has room for improvements. A more complex model would firstly take into account the vehicle side slip and the longitudinal slip of the wheel from which the velocity is measured. The fact that the length of the reconstructed trajectory compares well with the reference length of the circuit, suggests that these issues in the modelling might not be so relevant. Errors in the velocity due to the wheel longitudinal slip, e.g. when the car brakes, or to the vehicle side slip, are in fact localised and occur only over a small portion of the trajectory. Hence their influence on the length of the racing line is bound

to be small, but these local errors will affect the curvature estimate and their local effects will be felt along the entire racing line because of the double integration procedure.

Improving the kinematic model would make the procedure far more complex, as some of the data needed, e.g. the wheel longitudinal slip or the vehicle lateral velocity, are not available directly from the telemetry. Hence, for the purpose of the present work we shall go no further, as either the "error compensation" algorithm or the "stretching" algorithm will serve us sufficiently well later on.

Chapter 4.

The Driver Model

Driving a car involves using preview information of the road ahead. At the highest level of motor sport, rally driving is the ideal example to describe such concept. While the co-pilot reads to the driver well in advance concise information about the ideal path to follow, the driver, working like a control device, stores the information in his/her brain, as in a memory buffer, processes it to work out the necessary control input and then applies it to the car with sufficient advance to allow the vehicle to respond according to its natural dynamics. When it all goes according to plan, the car slides beautifully on a slippery forest track from one corner to the next as in a modern ballet, and fastest stage times are on hand. Sometimes, though, things do go wrong, and this reminds us that even highly skilled human beings can not always match ideal optimal controllers.

The mathematical model for the vehicle steering control, which is described in this section, derives from Linear Optimal Preview Control Theory (Kwakernaak and Sivan, 1972). An imaginary optical lever is projected forward from the vehicle. Path preview information is acquired by comparing the ideal path ahead with points along the optical lever. Then, the steering control input is evaluated simply as the weighted sum of these preview samples. If the vehicle operates in its linear region, the theory applies directly and the weighting coefficients may be derived straightforwardly (Valtetsiotis, 1999). However, the aim here is to join the driver model to a non-linear vehicle model. Therefore the theory is only used to guide the choice of the model parameters, which will be determined by heuristic methods.

Extensive work was done in the field of active vehicle suspensions by (Sharp, 1995, Sharp and Prokop 1995) using Linear Optimal Preview Control Theory. Particularly, two concepts that emerged from their work are applied here. The first is that linear optimal preview control involves diminishing returns, i.e. for any task, there is an amount of preview beyond which further preview is of no value to the controller. If such further preview were available, the controller would take no notice of it. The second concept is that the gain sequence associated with the preview error samples reflects the dynamics of the controlled plant, indicating the use of the preview control to motivate the system by operating preferentially on its eigenmodes. Thus, if the plant has a lightly damped mode, and the control bandwidth is sufficient to excite that mode, the optimal preview control gain sequence will show the oscillatory pattern of the eigenmode. On the other hand, if the plant is very well damped, the gain sequence will appear more like an exponential convergence on zero gain for large preview distance.

If this ideas are applied to the mathematical model of the vehicle steering control, the following picture emerges. A circuit racing car on dry tarmac will be a highly damped system. Thus, the reference scheme for the preview gain sequence of the steering control model has to be exponentially convergent to zero towards the end of the

preview distance. Qualitatively, this scheme describes well the driving style of circuit racing drivers, which tends to be progressive and smooth. Conversely, rally cars on low grip surfaces, e.g. gravel, mud or snow, will be a lightly damped system and the gain sequence has to show an oscillatory pattern. This, too, qualitatively mirrors the driving technique in rallies, where the drivers take advantage of the lightly damped vehicle lateral dynamics by sliding the car in order to control the path and achieve greater cornering speeds.

In vigorous manoeuvring, tyre forces saturate and imply non-linear vehicle behaviour. In this condition any increase of the control input would be ineffective. Furthermore, the tyre forces tend to fall far beyond saturation, so that an excessive control input would actually penalise the path following capabilities. In order to deal with this case, the basic linear structure of the steering control model is extended by including saturation functions. These are intended to accommodate the saturating non-linearity of the real vehicle by limiting the maximum control input in various conditions to pre-determined values, hence achieving more effective steer control when the vehicle is operated on the limit of its performance.

At this stage it is clear that the aim of the mathematical model for the vehicle steering control which is outlined here is to apply a control action based purely on the system dynamics in order to achieve effective path following. No attempt is made to mimic human driver behaviour. In fact, assuming that a highly skilled professional driver is capable of driving a car on its limits by applying a control which is very near the optimal one, the only thing relating to the real driver which is not included in the mathematical model is the natural lag that a human being would have from the moment that the preview information is acquired to the moment that the control is actually applied. With regard to preview control, though, this is irrelevant, since it would simply correspond to shifting the preview samples forward by as much as the expected human lag, and the optimal control would remain the same.

The driver model is finally completed by adding a strategy to control the vehicle longitudinal motion. At the simplest level, the vehicle forward velocity may be set equal to a constant value. Alternatively, the vehicle longitudinal control, i.e. the driving/braking torque, may simply be imposed. A more elaborate and convenient strategy, which will be considered here, consists in specifying a target longitudinal velocity along the intended path. The task is then to work out the vehicle longitudinal thrust which allows to closely follow the target velocity profile. It is important to notice that the lateral and longitudinal vehicle controls are treated here as being completely uncoupled. As a result, the intended manoeuvre may be infeasible, e.g. the longitudinal velocity or the vehicle thrust may be too high for a certain cornering radius. The simulation programme must then be able to handle these exceptional conditions, as we shall see.

The description of the driver model which is given in this section is intended to be general, regardless of the kind of vehicle model with which it is to be implemented. A specific application will be described in the next chapter and sufficient guidelines for setting the driver model parameters will be given. Various path following tasks will then be solved to demonstrate the effectiveness of the method.

4.1. THE PATH FOLLOWING TASK

Let the differential equations for a general vehicle model, representing the rate of change of its p state variables $\mathbf{x}$ with respect to the time t, be defined as follows:

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad \mathbf{x}(0) = \mathbf{x}_0 \qquad t \in [0, T] \qquad \text{Eq. 4.1}$$

Here, $\mathbf{u}(t)$ is a two-element vector function returning the vehicle lateral (i.e. the steer angle) and longitudinal (i.e. the driving/braking torque) controls. Since it is more convenient to describe the intended sequence of vehicle manoeuvres as a function of the path distance co-ordinate s, it is necessary to formulate the vehicle dynamic equations in such a way that they represent the rate of change of the states with respect to an increment of travelled distance ds along the ideal path. Let us define S_{CF} as the scaling factor relating the increment in time dt to the increment in travelled distance ds:

$$S_{CF} = \frac{dt}{ds} \qquad \text{Eq. 4.2}$$

Then, using Eq. 4.2, the vehicle model equations become:

$$\frac{d\mathbf{x}}{ds} = S_{CF} \cdot \mathbf{a}(\mathbf{x}(s), \mathbf{u}(s), s) = \bar{\mathbf{a}}(\mathbf{x}(s), \mathbf{u}(s), s) \quad \mathbf{x}(0) = \mathbf{x}_0 \quad s \in [0, S] \qquad \text{Eq. 4.3}$$

where S is the length of the ideal path that the vehicle has to follow.

The driver model is defined as an algebraic function which uses the path information together with a sub-set $\bar{\mathbf{x}}$ of the system states, and returns the vehicle controls $\mathbf{u}$ which yield effective path following. The required system states are: the longitudinal and lateral positions of the vehicle centre of gravity x_v and y_v, the vehicle yaw angle ψ_v, and the vehicle longitudinal and lateral velocities V_{vx} and V_{vy}:

$$\bar{\mathbf{x}} = \{x_v \quad y_v \quad \psi_v \quad V_{vx} \quad V_{vy}\}$$

The parameters used to describe the path information were defined in chapter 3, and they include the path curvature k_t, the tangent angle ψ_t, and the co-ordinates x_t and y_t measured from the reference axis system fixed in space. These parameters are taken as functions of the path length co-ordinate s. Road camber and elevation are not considered, but the extension would be straightforward if they were to be included in the vehicle modelling. Although the way that these data are organised and loaded in the simulation is arbitrary to a great extent, in the description of the driver model and its implementation we shall refer to the matrix $\mathbf{T}$ of trajectory parameters with the following structure:

$$\mathbf{T} = [\mathbf{s} \quad \mathbf{k}_t \quad \psi_t \quad \mathbf{x}_t \quad \mathbf{y}_t]$$

The first column specifies the distance from a chosen origin of a set of points along the ideal path, e.g. every five meters; the other columns contain the corresponding values of

the trajectory parameters. At any general location along the ideal path which is identified by the path co-ordinate s, the local trajectory parameters will be evaluated by means of linear interpolation. Finally, the path following task may formally be stated as follows:

$$\frac{d\mathbf{x}}{ds} = \bar{\mathbf{a}}(\mathbf{x}(s), \mathbf{u}(s), s) \qquad \mathbf{x}(0) = \mathbf{x}_0 \qquad s \in [0, S]$$
$$\textit{where}: \qquad \text{Eq. 4.4}$$
$$\mathbf{u}(s) = \mathbf{f}(\bar{\mathbf{x}}(s), \mathbf{T}, s)$$

4.2. TIME TO DISTANCE SCALING FACTOR

In this section we shall derive the expression for the time to distance scaling factor S_{CF} which was introduced with Eq. 4.2. The original formulation is due to (Allen, 1997) and the work presented here is a generalisation of his formula. The task is to express the increment ds of the distance travelled along a reference line corresponding to the increment dt in the manoeuvre time. For the case of the path following problem, the reference line is the ideal path that the vehicle must follow.

If the vehicle trajectory coincided exactly with the ideal path, the scaling factor would simply read:

$$S_{CF} = \frac{dt}{ds} = \frac{1}{V} \qquad \text{Eq. 4.5}$$

where V is the vehicle absolute velocity:

$$V = \sqrt{V_{vx}^2 + V_{vy}^2} \qquad \text{Eq. 4.6}$$

In the general case, though, the vehicle trajectory will depart from the reference line and the scaling factor must account for this condition.

Consider firstly figure 4.1(a). The relationship between the vehicle travelled distance ds_v and the time increment dt reads:

$$\frac{ds_v}{dt} = V = \sqrt{V_{vx}^2 + V_{vy}^2} \qquad \text{Eq. 4.7}$$

Since the vehicle heading direction differs from that of the tangent to the reference path at $s = s_a$, the effective travelled distance ds_1 measured on the reference path is obtained by projecting ds_v on the latter:

$$ds_1 = ds_v \cdot \cos(\nu - \psi_t) \qquad \text{Eq. 4.8}$$

A further condition has to be accounted for. When the vehicle is cornering, the distance that it has to travel depends on whether it takes an inner line or an outer line with respect to the reference path. Referring to figure 4.1(b), the relationship between the

increment of the travelled distance in the direction of the tangent to the path ds_1, and the corresponding increment of the path length ds reads:

$$\frac{ds_1}{ds} = \frac{(r_t - d)}{r_t} = \left(1 - \frac{d}{r_t}\right) \qquad \text{Eq. 4.9}$$

Here, r_t is the corner radius and d is the distance of the vehicle centre of gravity from the intended path. Knowing the position of the car in the absolute reference axis system fixed in space (x_v, y_v), and the co-ordinate (x_t, y_t) and the orientation ψ_t of the corresponding point on the reference path, the distance d is measured on the perpendicular to the reference path through s_b and is calculated with the following expression:

$$d = (y_v - y_t)\cdot\cos(\psi_t) - (x_v - x_t)\cdot\sin(\psi_t) \qquad \text{Eq. 4.10}$$

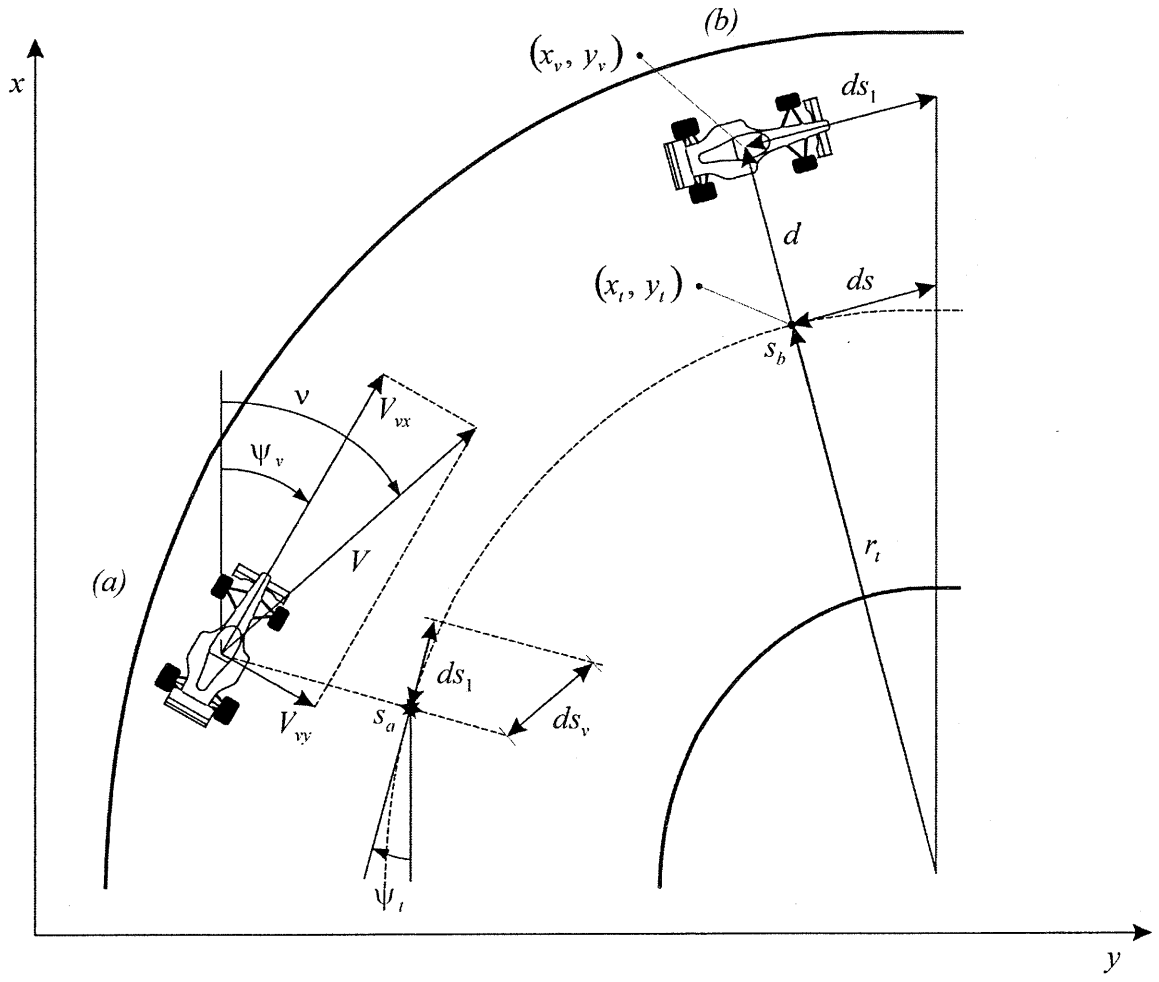


Figure 4.1 Vehicle travelled distance projected onto the reference line.

Combining Eqs. 4.7, 4.8 and 4.9 the scaling factor required to determine the state derivatives with respect to the new independent variable s may be calculated as follows:

$$\frac{d\mathbf{x}}{dt} = \frac{d\mathbf{x}}{ds} \cdot \frac{ds}{ds_1} \cdot \frac{ds_1}{ds_v} \cdot \frac{ds_v}{dt} = \frac{d\mathbf{x}}{ds} \cdot \frac{V \cdot \cos(\nu - \psi_t)}{(1 - d/r_t)} \qquad \text{Eq. 4.11}$$

Hence:

$$\frac{d\mathbf{x}}{ds} = \frac{d\mathbf{x}}{dt} \cdot \frac{(1 - d/r_t)}{\sqrt{V_{vx}^2 + V_{vy}^2} \cdot \cos(\nu - \psi_t)} = \frac{d\mathbf{x}}{dt} \cdot S_{CF} \qquad \text{Eq. 4.12}$$

By assuming that $V_{vy} << V_{vx}$, the following approximations hold:

$$\nu = \arctan\left(\frac{V_{vy}}{V_{vx}}\right) + \psi_v \cong \left(\frac{V_{vy}}{V_{vx}}\right) + \psi_v \qquad \text{Eq. 4.13}$$

$$\sqrt{V_{vx}^2 + V_{vy}^2} \cong V_{vx} \qquad \text{Eq. 4.14}$$

By substituting Eqs. 4.13 and 4.14 in Eq. 4.12 we obtain:

$$S_{CF} = \frac{(1 - d/r_t)}{V_{vx} \cdot \cos\left(\frac{V_{vy}}{V_{vx}} + (\psi_v - \psi_t)\right)} \qquad \text{Eq. 4.15}$$

Following the assumption that the vehicle lateral velocity is much smaller than the vehicle longitudinal velocity, the cosine function in the denominator of Eq. 4.15 may be approximated with the following expression:

$$\begin{aligned} &\cos\left(\frac{V_{vy}}{V_{vx}} + (\psi_v - \psi_t)\right) = \\ &\cos\left(\frac{V_{vy}}{V_{vx}}\right) \cdot \sin(\psi_v - \psi_t) - \sin\left(\frac{V_{vy}}{V_{vx}}\right) \cdot \cos(\psi_v - \psi_t) \cong \\ &\sin(\psi_v - \psi_t) - \frac{V_{vy}}{V_{vx}} \cdot \cos(\psi_v - \psi_t) \end{aligned} \qquad \text{Eq. 4.16}$$

Finally, the substitution of Eq. 4.16 into Eq. 4.15 yields the final expression for the scaling factor:

$$S_{CF} = \frac{(1 - d/r_t)}{V_{vx} \cdot \cos(\psi_v - \psi_t) - V_{vy} \cdot \sin(\psi_v - \psi_t)} \qquad \text{Eq. 4.17}$$

4.3. STEER CONTROL THROUGH PATH PREVIEW

4.3.1. Linear control scheme

Figure 4.2 shows the scheme of the multiple preview point steer angle controller. At a generic position s along the trajectory the driver monitors the path ahead by projecting an optical lever forward, up to a distance L_p which is chosen by the analyst to represent the extent of the preview available, giving a preview time T_p, hence making this distance a function of the vehicle velocity:

$$L_p = V_{vx} \cdot T_p \qquad \text{Eq. 4.18}$$

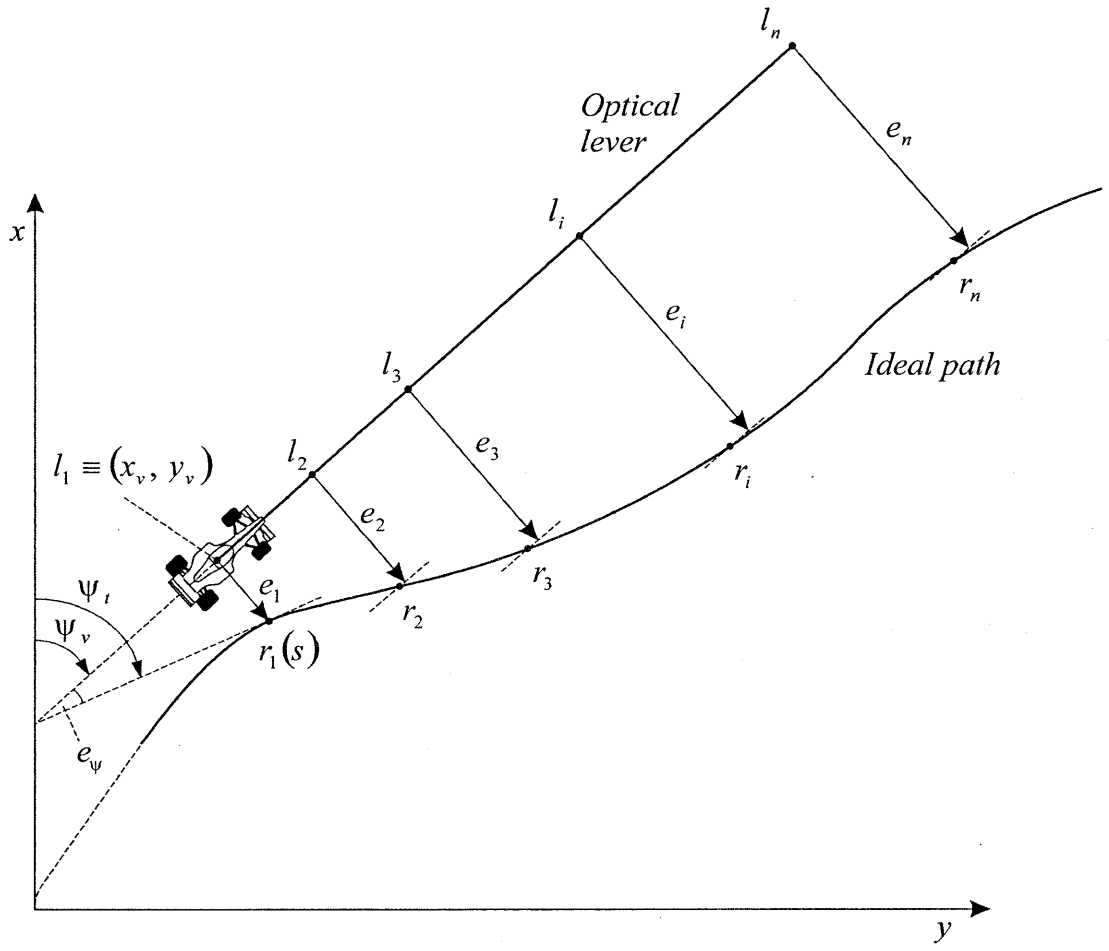


Figure 4.2 Driver model scheme.

A set of n points $l_1, l_2, ..., l_n$ is then chosen on the optical lever, the first point coinciding with the vehicle centre of gravity whose co-ordinates are $x_v(s)$ and $y_v(s)$. The vehicle attitude angle $\psi_v(s)$ determines the orientation of the optical lever and allows the calculation of the positions of the other n-1 points. Let us next define the vector Δ_L which specifies the relative distances of the points on the optical lever from the origin $x_v(s)$ and $y_v(s)$. Its i^{th} component reads:

$$\Delta_{Li} = \frac{\overline{l_i l_1}}{L_p} \qquad i = 1 \div n \qquad \text{Eq. 4.19}$$

Note that according to the Eq. 4.19, Δ_{L1} will be equal to 0. The vector of preview distances $\mathbf{s_P}$ may then be evaluated as follows:

$$\mathbf{s_P} = L_p \cdot \mathbf{\Delta_L} \qquad \text{Eq. 4.20}$$

and, finally, the co-ordinates of the points $l_1, l_2, ..., l_n$ read:

$$\begin{cases} \mathbf{x_L} = x_v(s) + \mathbf{s_P} \cdot \cos(\psi_v(s)) \\ \mathbf{y_L} = y_v(s) + \mathbf{s_P} \cdot \sin(\psi_v(s)) \end{cases} \qquad \text{Eq. 4.21}$$

From each of these points, the lateral off-set from the optical lever to the corresponding point $r_1, r_2, ..., r_n$ on the ideal path is measured and used as input to the steer angle controller. The strategy to identify the reference points on the path ahead is chosen in order to simplify the calculations as follows: the distance of the generic point r_i measured along the ideal path from the point r_1 is taken equal to the distance between the corresponding point l_i and l_1 measured on the optical lever. Hence, the vector of preview distances $\mathbf{s_P}$ is added to the current path co-ordinate s to obtain the positions of the $r_1, r_2, ..., r_n$ points along the path:

$$\mathbf{s_R} = s + \mathbf{s_P} \qquad \text{Eq. 4.22}$$

Then, the co-ordinates of these points in the reference axis system are evaluated using the matrix of path data $\mathbf{T}$, defined in § 4.1, as a one dimensional look-up table with the vector $\mathbf{s_R}$ as entry value:

$$\begin{cases} \mathbf{x_R} = \text{lin\_interp}(\mathbf{T}(:,1), \mathbf{T}(:,4), \mathbf{s_R}) \\ \mathbf{y_R} = \text{lin\_interp}(\mathbf{T}(:,1), \mathbf{T}(:,5), \mathbf{s_R}) \end{cases} \qquad \text{Eq. 4.23}$$

Here, the notation $(:,n)$ indicates the n^{th} column of the matrix. By following this scheme, the $r_1 \div r_n$ points will not lie exactly on the direction perpendicular to the optical lever through the $l_1 \div l_n$ points and the error will be dependent on the complexity of the trajectory geometry. However, this does not seem to have any significant influence on the robustness of the driver model and this simplified approach is computationally advantageous. Having identified the set of co-ordinates ($\mathbf{x_L}$, $\mathbf{y_L}$) and ($\mathbf{x_R}$, $\mathbf{y_R}$) in the reference axis system fixed in space, the lateral off-sets $e_1, e_2, ..., e_n$ in figure 4.2 are evaluated by referring the position of each of the r_i points to a local axis system whose origin is set on the corresponding l_i point on the optical lever and the local axes are oriented as those of the vehicle local reference axis system, i.e. the x-axis is directed forward and the y-axis is directed to the right hand side of the vehicle. The expression for the lateral off-sets then reads:

$$e_i = (y_{Ri} - y_{Li}) \cdot \cos(\psi_v) - (x_{Ri} - x_{Li}) \cdot \sin(\psi_v) \qquad i = 1 \div n \qquad \text{Eq. 4.24}$$

Of these n control inputs, $e_2 \div e_n$ are preview path information, whereas e_1 represents the vehicle lateral off-set from the current intended position on the path and is the first of the two state feedback inputs used by the driver model. The other state feedback signal is taken as the difference between the vehicle attitude angle and the tangent angle of the intended path at the current position s, which is identified by means of linear interpolation as above:

$$\psi_t = \text{lin\_interp}(\mathbf{T}(:,1), \mathbf{T}(:,3), s) \qquad \text{Eq. 4.25}$$

and:

$$e_\psi = \psi_t - \psi_v \qquad \text{Eq. 4.26}$$

Finally, introducing the set of control gains K_1, K_2, ..., K_n and K_ψ, the steer angle is evaluated as a linear combination of the control inputs:

$$u_{st} = K_\psi \cdot e_\psi + K_1 \cdot e_1 + \sum_{i=2}^{n} K_i \cdot e_i \qquad \text{Eq. 4.27}$$

4.3.2. Non-linear control scheme with saturation functions

The linear control law of Eq. 4.27 would correspond with linear tyre forces, but when a vehicle is operated on the limit of its performance envelope, the tyre forces saturate and any increase of the control action will be ineffective. For example, if the vehicle model were asked to follow an impossible path so that its trajectory were to depart from that path significantly, the control inputs would grow a great deal and Eq. 4.27 would return unreasonable values for the steer angle, causing the tyre forces to fall far beyond saturation. The best strategy in such conditions would be to limit the steer angle in order to keep the tyre as closely as possible to saturation, therefore maximising the control forces, until the vehicle would eventually return to the original path. In order to deal with such cases, the driver model structure is extended by including saturation functions as shown in figure 4.3. This structure is designed to handle different limit behaviour of the vehicle.

When an understeer vehicle reaches its lateral limit, its front tyres saturate first and it tends to run on a trajectory wider than the ideal path. In this situation the vehicle attitude error gives a much smaller contribution to the generation of the steer angle compared to the other preview and state feedback lateral off-set inputs. Therefore a saturation function is placed to limit the value of the sum of the contributions of all the lateral preview and state feedback off-sets. This prevents the front tyres from working too far beyond saturation, hence maximising the lateral control force to enhance the capability of the driver model to return to the original intended path. Conversely, when the rear tyres of a vehicle reach saturation first, e.g. because of its oversteer nature, or while cornering under driving or braking, the attitude error control gives the most significant contribution. In this condition the steer input required may be much larger than in the previous case of the understeer vehicle. However, to prevent the front wheels from steering beyond the physical limit of a real car, an overall saturation function is applied to the output of the driver model.

If the vehicle slides temporarily away from the intended line, e.g. because of understeer, when trying to regain the original course its trajectory may cross the ideal path with a high angle of incidence. In this condition the simple linear structure of Eq. 4.27 plus the global saturation functions would return a steep shift of the steer angle from the maximum to the minimum value and vice versa, hence inducing some oscillations before the vehicle may recover the original course on the ideal path. To further improve the driver model response in such extreme manoeuvres, the contribution to the steer angle generation from each of the preview and lateral state feedback inputs is limited by applying individual saturation functions. This scheme has a more progressive action in such extreme conditions and improves the capability of the driver model to deal with limit manoeuvres.

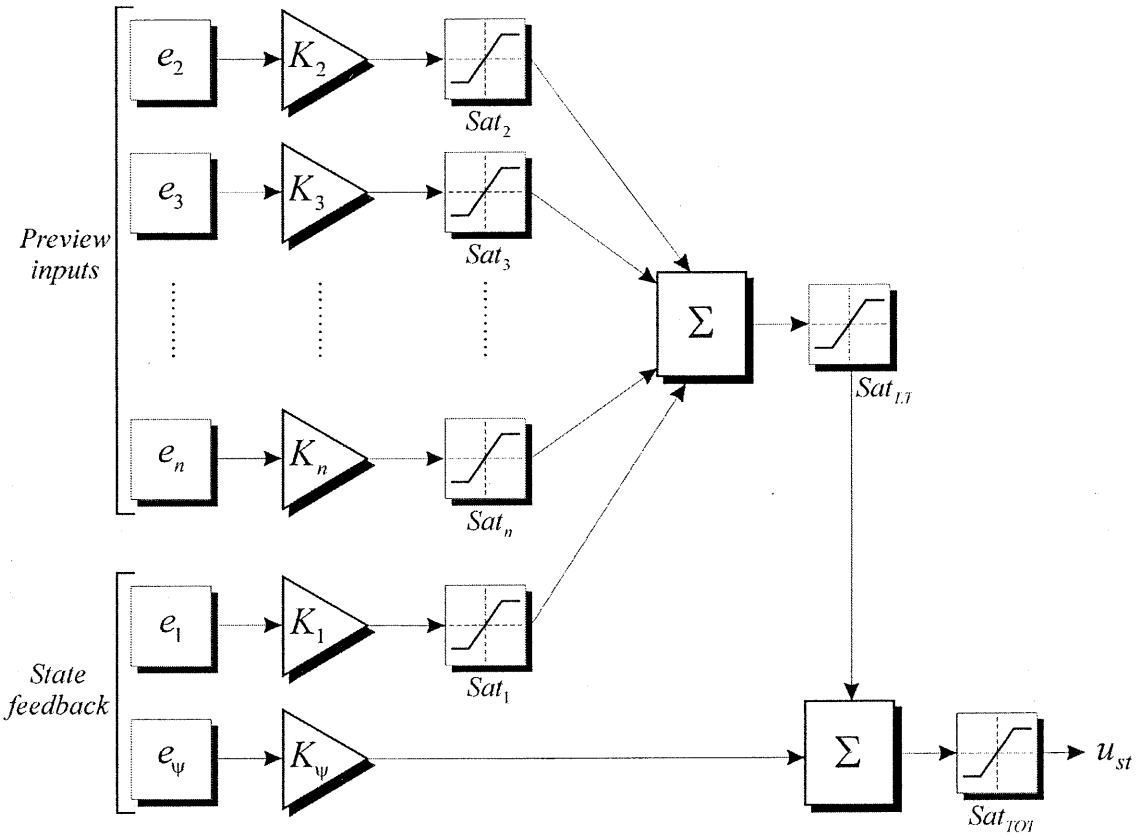


Figure 4.3 Steer angle control scheme with saturation functions.

Figure 4.3 shows then the final scheme of the multiple preview points steer angle control model. In order to implement this model, one needs to set the n+1 control gains associated with the n lateral off-sets e_1, e_2, ..., e_n and the vehicle attitude off-set e_ψ. Furthermore, the model includes n+2 more parameters, the individual saturation levels Sat_i and the global ones, Sat_{LT} and Sat_{TOT}. Finally, also the structure of the preview model must be defined, and this includes setting the preview time T_p, the number of preview points on the optical lever and their relative positions from its origin.

4.4. SPEED CONTROL SCHEME

Let $V_t(s)$ be the target velocity profile that the vehicle must follow while proceeding along the intended path. The task is to determine the vehicle longitudinal thrust which is required in order to closely follow the target velocity by solving a simple one dimensional inverse dynamics problem. Consider the vehicle travelling with a velocity V_{vx} at a generic position s along the ideal path. From the current position the driver model reads the target velocity from the ideal velocity profile at a short distance Δs ahead, $V_t(s+\Delta s)$. The constant forward acceleration which would be required to reach the target velocity may be evaluated as follows:

$$a_{vx}(s) = \frac{[V_t(s+\Delta s) - V_{vx}(s)]}{\Delta s} \cdot \frac{1}{S_{CF}} \qquad \text{Eq. 4.28}$$

Here, the scaling factor S_{CF} is required to obtain the time derivative of the velocity. Using the values of the vehicle forward velocity and acceleration, the longitudinal force F_{vx} that is required to maintain the ideal forward motion may be estimated by observing that the main contributions are due to inertial effects and to the aerodynamic drag. In general terms we may write:

$$F_{vx}(s) = f_1(a_{vx}(s)) + f_2(V_{vx}^2(s)) \qquad \text{Eq. 4.29}$$

The first term in Eq. 4.29 includes the contributions from all the inertia forces that are accounted for in the vehicle model, e.g. the inertia of the vehicle body, the inertia of the rotating masses at the front and rear axles, the inertia associated with the moving parts of the engine, etc. The second term represents the aerodynamic drag and it depends on the square of the forward vehicle velocity. The longitudinal thrust evaluated with Eq. 4.29, though, is only an approximation of the ideal value which would be required to perfectly match the target velocity, as other small effects like the drag induced by the steering wheels in cornering are not included. However, this is acceptable and yields only a little loss in accuracy, since the speed control scheme works as a simple feedback control and is therefore self correcting. Finally, once the longitudinal thrust has been evaluated, its application to the vehicle depends on how the vehicle model has been conceived. Therefore we shall not go further here, and the discussion is postponed to the application which will be presented in the next chapter.

4.5. EXCEPTION HANDLING: INFEASIBLE MANOEUVRES

Although the driver model has proved to be capable of controlling the vehicle in many extreme conditions, as will be demonstrated in the next section, decoupling the vehicle longitudinal control from the lateral control implies the possibility of infeasible manoeuvres. Exceptional conditions may occur when the ideal velocity profile is impossible, in view of tyre force saturation. When the path following task is compromised, it is convenient to have a method to detect such a condition and to stop the simulation, returning the results which allow one to understand what has happened. In order to do this, the value of the scaling factor S_{CF} is monitored during the

simulation. If the vehicle follows closely the ideal path, the value of the scaling factor is almost equal to the inverse of the forward velocity:

$$S_{CF} = \frac{dt}{ds} \cong \frac{1}{V_{vx}} \qquad \text{Eq. 4.30}$$

But if the vehicle drifts away from the ideal path or tends to spin off, Eq. 4.30 does not hold any longer. In the limit case that the vehicle trajectory becomes perpendicular to the ideal path, the increment of travelled distance *ds* along the latter would be zero for any variation of the vehicle states, the scaling factor would become infinite and the equations of motion of the vehicle singular. If, instead, the vehicle trajectory lies much on the inside or the outside of the ideal path through a corner, by virtue of Eq. 4.9 the scaling factor will be smaller or greater than the value predicted by Eq. 4.30 respectively.

Exception handling is realised by allowing a certain range of variation of the scaling factor as a measure of good path following. When the scaling factor exceeds the set limits the simulation is stopped and the results returned up to the last vehicle position, in order to detect what caused the driver model to fail. The range of variation for the scaling factor which is normally allowed is:

$$0.25 \cdot \frac{1}{V_{vx}} \leq S_{CF} \leq 4 \cdot \frac{1}{V_{vx}} \qquad \text{Eq. 4.31}$$

4.6. CONCLUSIONS

A new structure for a steering controller for road vehicles has been devised. The controller takes in sample values of preview path errors, lateral position error and attitude error and converts the observations into a steering wheel angle control. The structure is based in Linear Optimal Preview Control Theory but it has been developed to deal with non-linear vehicle operation arising from the inevitable tyre force saturation in vigorous manoeuvring. This implies that the formal theory may not be applied directly. The fundamental concepts which may be used to guide the setting of the model parameters have been highlighted and the final values have to be chosen by heuristic methods. The task to determine all the driver model parameters without a formal procedure may appear at first hardly feasible. However, as we shall see in the application presented in the next section, at least for the case of a circuit racing car on dry tarmac the reference model adopted for the steering control, i.e. exponentially decaying control gain patterns, provides a viable approach to set up the model.

In view of a more general application, though, we may observe that the steering control structure shown in figure 4.3 is similar to that of a neural network. It seems therefore likely that in future the parameter tuning process will be achievable by standard neural network learning procedures (Tarassenko, 1998). Another issue for the future development of the model is the coupling of the vehicle lateral and longitudinal control into a single structure, which would have the potential to improve the path following capabilities when the vehicle is operated on its limit performance.

Very recently, after the work which is presented here was completed, an important

contribution was made by (Valtetsiotis, 1999). He solved the formal problem by linking the multiple preview control driver model to a linear vehicle model. The control gains associated with preview and state feed-back information were determined by applying the theory directly to a linear optimal control problem whose objective was to minimise the vehicle lateral and angular off-sets from an ideal path. Figure 4.4 was generated using the software developed by Valtetsiotis for his research. The typical control gain patterns which yield effective path following at four different velocities and for two very different vehicle configurations are shown. The first graph refers to a circuit racing car on dry tarmac, which is characterised by low polar moment of inertia, long wheelbase and high front and rear cornering stiffnesses. The second graph refers instead to a typical rally car, which, conversely, is characterised by higher polar moment of inertia, shorter wheel base and much lower cornering stiffnesses. The results indicate firstly that the amount of preview distance needed depends on the vehicle velocity as well as on the vehicle natural dynamics. In other words, a rally car on a low grip surface will have a much slower response compared to a circuit racing car, hence requiring longer preview. The oscillatory nature of the control required by the rally car is also evident in comparison with the control for the circuit race car, hence supporting the assumptions made above.

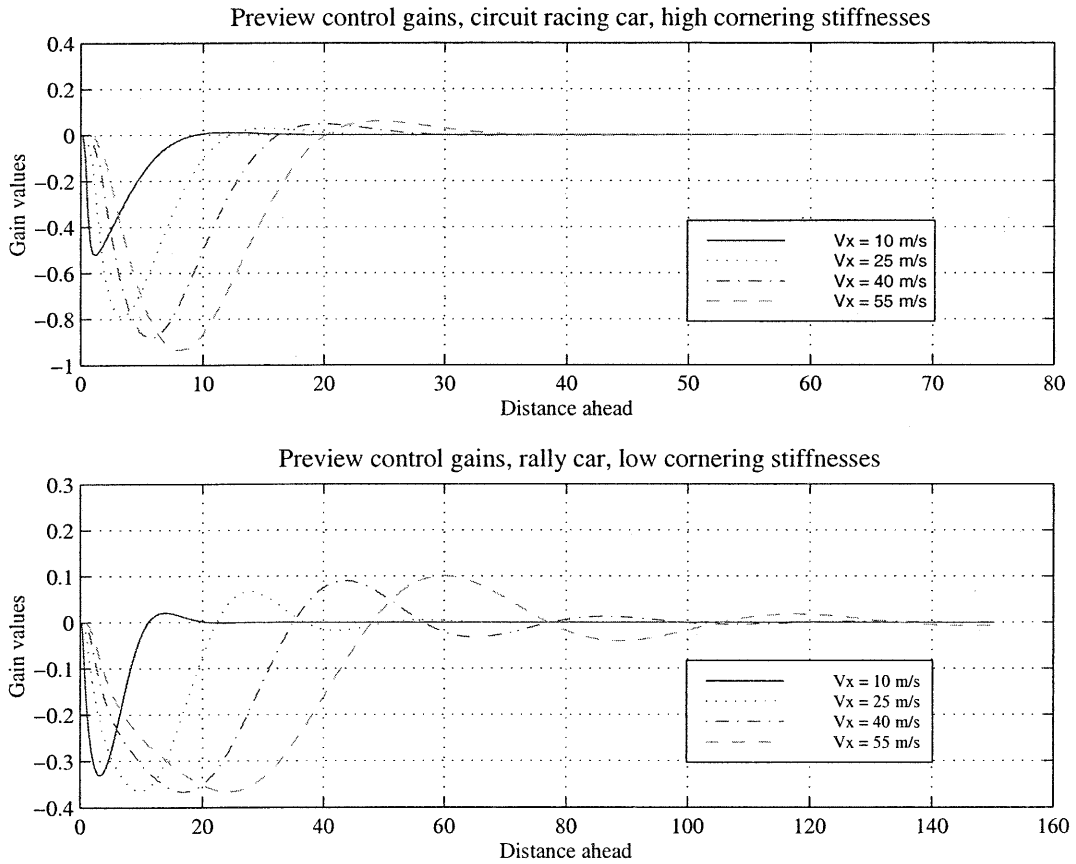


Figure 4.4 Control gain patterns for two different car configurations.

Finally, in the linkage of the steering controller to a vehicle dynamics simulation model, the transformation of the problem from time domain to distance domain is seen

as important. The transformation method used is in advance of previous work and it has contributed usefully to the outcome. The main advantage is that the vehicle position may be compared directly with the corresponding intended position along the ideal path. It has also allowed a convenient check on the health of a simulation run to be incorporated into the calculations, mitigating against wasted runs.

Chapter 5.

Path Following Results

The driver model described in the previous section is joined here to a vehicle dynamics model of a current Formula One car. Its path tracking performance is demonstrated by setting up various tasks involving moderate manoeuvring as well as racing speeds. Each task consists of a specified ideal path to follow with an imposed target velocity.

The vehicle dynamics model is described in detail in Appendix A. However, as its structure influences the setting up of the driver model, a brief outline of the vehicle model adopted will be given firstly. Then, the parameters for the mathematical model of the steering control are derived. Also the longitudinal control scheme, which specifically depends on how the vehicle model is conceived, is described in detail. Finally, the results for various path following tasks are presented, firstly using a double lane change manoeuvre specifically designed to simulate moderate "*g*" vehicle operations, and then using the racing lines which were obtained in Chapter 3 to simulate race car level operations.

5.1. VEHICLE MODEL OUTLINE

The vehicle is represented with a seven degrees of freedom model. The chassis is treated as a rigid body with three degrees of freedom, the yaw angle and the lateral and longitudinal displacements. The wheels are rigidly attached to the body, allowing only the rotation of each wheel relative to the vehicle chassis about its spin axis. Therefore, they add four more degrees of freedom to the system. Although the suspensions are not included in the modelling, the roll axis position, the roll stiffness distribution, the mass centre height and the track width are specified. These parameters are sufficient to derive approximate expressions for the lateral and longitudinal load transfers corresponding to a constant acceleration of the mass centre (Dixon, 1996).

A simple representation of the in-plane aerodynamic forces is employed by assuming constant drag and lift coefficients. Since the vehicle model does not include pitch and heave degrees of freedom, the dependency of the aerodynamic forces on the vehicle front and rear ride heights is not accounted for. The force system is then described by the aerodynamic drag, which is applied at the height of the vehicle centre of gravity, and the aerodynamic lift, whose centre of application is assumed to be the same for all speeds and is determined by specifying the down force distribution between the front and the rear axles. No aerodynamic moment needs then to be specified.

Realistic tyre lateral and longitudinal forces are introduced using the Magic Formula Tyre Model which features the use of weighting functions to account for combined slip conditions (Pacejka and Besselink, 1997). The tyre slip quantities are

evaluated with the assumption of small angles (i.e. for a generic angle α it is assumed that $\cos(\alpha) \cong 1$ and $\sin(\alpha) \cong \alpha$) and they are referred to the centre of the contact area of each wheel. Static wheel camber angle settings are also accounted for. Finally, the wheel vertical loads are evaluated accounting for the vehicle static weight distribution, the aerodynamic down-force distribution and the longitudinal and lateral load transfers when the vehicle is driving, braking and cornering.

The vehicle lateral control variable is the steer angle applied to the front wheels. A parallel steer geometry is considered for the steering system, i.e. the right and left front wheels assume the same steer angle. A single control variable is defined for the longitudinal control. It is assumed that this variable represents the throttle aperture when it is positive, or a portion of the maximum braking torque available when it is negative. When driving, firstly the gear ratio is selected based upon the vehicle forward velocity. Then, the engine rotational velocity is computed using the mean rotational velocity of the rear wheels and the total gear ratio. This value is used together with the throttle input in order to read the engine output torque from an experimental engine map. Using the gear ratio, the torque available to the driven wheels is computed. The individual torque applied to each wheel is finally evaluated taking into account the effect of the engine moment of inertia and the torque transfer due to a limited slip differential. The simple model used to compute the torque transfer is based on the Salisbury type differential (Milliken and Milliken, 1995). In this differential clutch packs are used to progressively lock the two output shafts together. The axles of the differential pinions rest against wedges which are spread by the input torque, so that they apply a pressure to the clutch packs which is proportional to the input torque. Furthermore, the wedges may have different angles to give different characteristics in driving and in overrun. The amount of torque transfer may then be evaluated as a constant portion of the input torque. Two different torque gains G_{D2_D} and G_{D2_O} are used to represent the different characteristics of the differential in driving and in overrun respectively. Also a constant off-set G_{D1} is included to account for the natural friction in the differential. According to the SAE sign conventions, the driving torque is negative and the braking torque is positive. Hence, if T_D is the input torque to the differential cage, the torque transfer reads:

$$\begin{aligned} T_T &= -G_{D1} + G_{D2\_D} \cdot T_D \quad \text{if} \quad T_D < 0 \quad (\textit{driving}) \\ T_T &= -G_{D1} - G_{D2\_O} \cdot T_D \quad \text{if} \quad T_D > 0 \quad (\textit{overrun}) \end{aligned} \qquad \text{Eq. 5.1}$$

Eq. 5.1 always returns a negative value for the torque transfer. The direction of the torque transfer which is actually applied to the rear wheels depends on the sign of their differential velocity[1]:

$$\Delta T = T_T \cdot \text{sign}(\omega_{RR} - \omega_{LR}) \qquad \text{Eq. 5.2}$$

When braking, the longitudinal control variable simply factors the maximum braking torque available. Then the actual braking torque is shared among the front and rear axles using constant coefficients. The engine brake effect is then added to the rear

[1] From now on, the subscripts *LF*, *RF*, *LR* and *RR* are used to indicate left front, right front, left rear and right rear wheels respectively. For more details see the section "Notations".

axle. Finally, the brake torque is split evenly between the front wheels, while the effect of the limited slip differential is accounted for at the rear axle.

5.2. STEER ANGLE CONTROL: PARAMETER SETTING AND TUNING

The mathematical model of the steering control which was described in chapter 4 is joined here to a vehicle model representing a contemporary Formula One race car. Such vehicle when running on a high grip surface (e.g. dry tarmac) will be a highly damped system. Therefore, in view of the theory outlined earlier, the reference scheme for the driver model preview gain pattern which is adopted is that with the gain sequence exponentially convergent to zero towards distant preview samples. Figure 5.1 visualises this scheme by showing qualitatively the contribution to the steer angle control from the various preview control inputs.

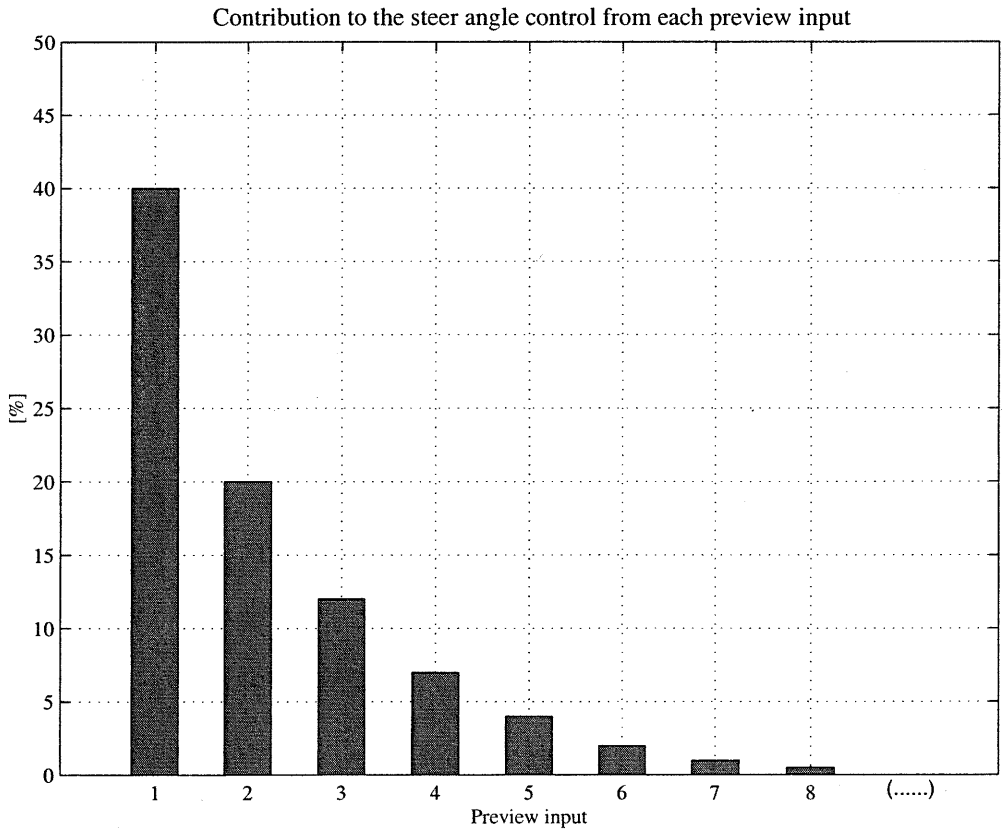


Figure 5.1 Relative contributions to the steer angle from preview path errors.

An important feature of the steering control model is that it uses state feedback together with preview information. The state feedback control gains are important in increasing the system stability. Especially the vehicle attitude error feedback control improves the performance of the driver model for vehicles made artificially unstable by strong oversteer. However, the contribution of the attitude error gain tends to reduce the steer angle when cornering normally, because of the natural attitude of the car in generating slip angles at the rear wheels. Hence, the vehicle runs slightly off the ideal path so that the contribution of the lateral off-set feedback control cancels out the

negative contribution of the attitude error control.

Before we may proceed with the model parameter setting, the initial structure of the driver steering control must be defined in terms of preview time, the number of preview points and their positions along the optical lever. For the current implementation the preview time has been set equal to one second and 8 points have been placed on the optical lever, including the one which accounts for the state feedback lateral off-set. Although it may seem to be natural to evenly space these points along the optical lever, a different scheme is employed here in view of the concept of diminishing returns, which consists in gathering the preview points towards the vehicle centre of gravity. At this stage, though, such a strategy has not played a significant role, but to further refine the driver model the distribution of the preview points along the optical lever may be varied in order to enhance computational efficiency by using the least number of points which returns sufficient path tracking accuracy.

The setting up of the driver model parameters involves two phases. Firstly, a set of initial values for the gains is derived by proceeding as follows. Several open-loop simulations are performed by setting constant values of the steer, throttle and brake controls each time. Then, using the values of the vehicle state variables, the optical lever is projected from several fixed positions on the simulated trajectories in order to read what would be the input to the controller. Once the preview control inputs corresponding to a certain steer angle are known, the gains to generate such value for the steer angle may be estimated on the basis of the exponential diminishing returns scheme. For example, the contribution of the first preview input (e_2 in figure 4.2) may be set equal to the 40 % of the total steer angle, that of the second preview point equal to the 20 %, then 12 %, 8 % end so on, as was qualitatively described in figure 5.1. Less information may be given about the initial setting of the state feedback gains, though. A good estimate to begin with is to set them equal to the numerical value of the first preview gain.

The second phase consists in tuning the set of parameters of the driver model in order to minimise the path tracking error and to improve the vehicle control in limit manoeuvres. Increasing the attitude error gain usually improves the control of unstable vehicles. As a consequence, this may result in a larger path tracking error since the contribution of the attitude error control tends to reduce the steer angle when the vehicle is cornering normally as was explained above. The state feedback lateral off-set control gain may be used to compensate this effect. The setting of the saturation levels may be done at this stage, since their influence is relevant only in extreme conditions. The total saturation is usually set equal to the maximum steer angle achievable by the real car (limited by steering stops in the mechanism) in order to prevent the driver model from returning absurd values. The saturation function which is applied to the sum of the contributions of all the preview and state feedback lateral off-sets is more relevant when controlling understeer vehicles, where the lateral path tracking error is much more significant than the vehicle attitude error. In such conditions the steer angle should be limited to a value which sets the front tyres working at a slip angle which is as close as possible to that at lateral saturation, in order to maximise the control force. However, the proper value varies depending on the kinematic conditions, tyre vertical loads, etc., hence the choice will be made in order to compromise the driver response in several different manoeuvres. Finally, the individual saturation levels have the function of smoothing the action of the driver model when trying to regain the ideal path after large excursions from the intended line, e.g. in case of strong understeer. As a general rule it

may be indicated that the contribution from each of the lateral off-set controls should not be greater than the 10 ÷ 20 % of the maximum steer angle allowed. Table 5.1 reports all the values of the parameters of the driver model currently implemented. It has been found necessary to change these values infrequently.

DRIVER MODEL PARAMETERS			
LATERAL OFF-SET CONTROL PARAMETERS			
Index	Relative Position	Gain [deg/m]	Saturation level [deg]
i	Δ_{Li}	K_i	Sat_i
1	0	10	±1
2	0.1	10	±2
3	0.2	6	±2
4	0.3	2	±2
5	0.4	0.8	±2
6	0.6	0.16	±1
7	0.8	0.04	±1
8	1	0.01	±1

ATTITUDE ERROR CONTROL GAIN, $K\psi$	30 [deg/rad]
TOTAL LATERAL OFF-SET CONTROL SATURATION LEVEL, Sat_{LT}	±10 [deg]
GLOBAL SATURATION LEVEL, Sat_{TOT}	±16 [deg]

Table 5.1 Steering control model parameters.

5.3. SPEED CONTROL: INVERSE LONGITUDINAL DYNAMICS

The task of the longitudinal vehicle control is to determine the longitudinal thrust which is required in order to closely follow a target velocity profile which is specified along the intended path. The general procedure described in §4.4 uses the estimate of vehicle longitudinal acceleration and the vehicle current velocity to evaluate the longitudinal thrust on the basis of the forces which the vehicle model includes. Then, the longitudinal thrust must be applied to the vehicle model. For the present application this implies to determine the equivalent driving or braking torque at the wheels and this must be done according to the law implied by the models of the powertrain and the braking system. Figure 5.2 shows the reference scheme for the longitudinal vehicle control. The values for the torque applied to each wheel are determined in such a way that the sum of the longitudinal tyre contact forces which balance the applied torque equals the longitudinal thrust:

$$F_{LF} + F_{RF} + F_{LR} + F_{RR} = F_{vx} \qquad \text{Eq. 5.3}$$

Let $V_t(s)$ be the target velocity profile specified as a function of the travelled distance. Consider the vehicle travelling with a velocity V_{vx} at a generic position s along the intended path. As was described in the previous chapter, we shall begin by

evaluating the constant forward acceleration which would be required to reach the target velocity at a short distance Δs ahead $V_t(s+\Delta s)$:

$$a_{vx}(s) = \frac{[V_t(s+\Delta s) - V_{vx}(s)]}{\Delta s} \cdot \frac{1}{S_{CF}} \qquad \text{Eq. 5.4}$$

The fact that Eq. 5.4 uses the difference between the target velocity ahead and the actual vehicle velocity implies that the longitudinal control scheme operates as a simple feedback control.

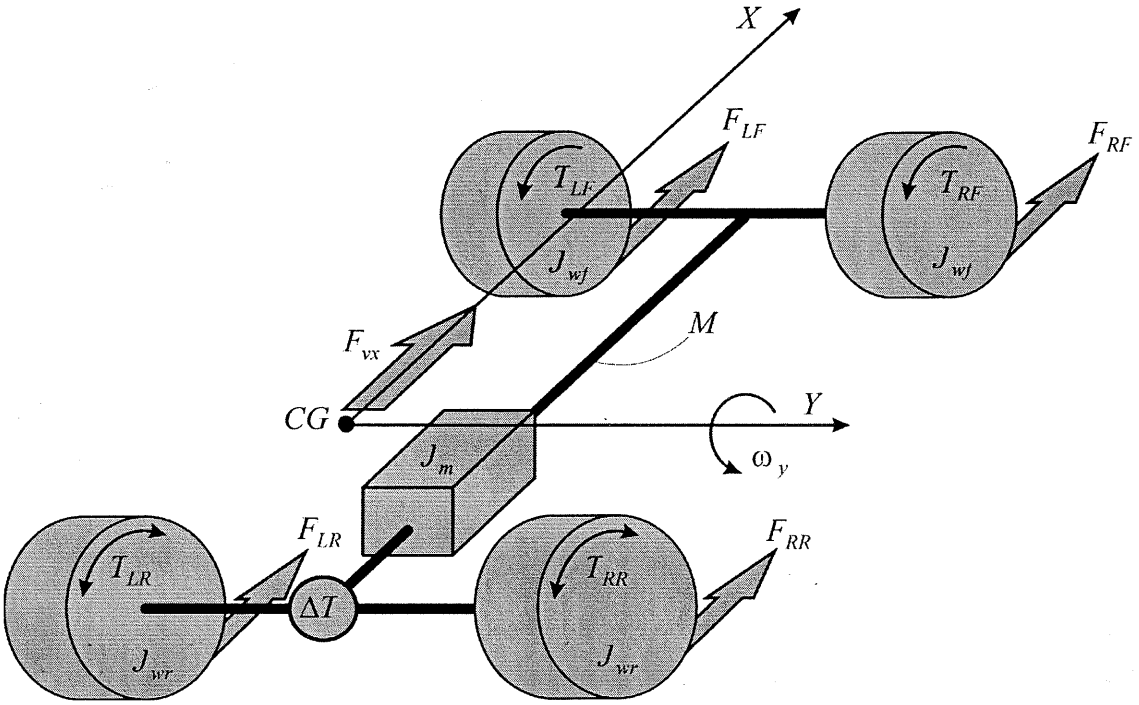


Figure 5.2 Vehicle longitudinal control scheme.

Next, the longitudinal thrust $F_{vx}(s)$ which is required to maintain the ideal forward motion is estimated accounting for the inertia of the vehicle, the inertia of the rotating masses, i.e. the wheels and the engine, and the aerodynamic drag. Neglecting the individual wheel longitudinal slip, the rotational acceleration of the wheels may be approximated simply by dividing the vehicle forward acceleration by the front and rear wheel radii, R_f and R_r. Hence, we may write:

$$F_{vx}(s) \cong M \cdot a_{vx}(s) + 2 \cdot J_{wf} \cdot \frac{a_{vx}(s)}{R_f} + \left(2 \cdot J_{wr} + J_m \cdot G_R^2\right) \cdot \frac{a_{vx}(s)}{R_r} + \frac{1}{2} \cdot \rho \cdot S_f \cdot C_x \cdot V_{vx}^2 \qquad \text{Eq. 5.5}$$

In Eq. 5.5, M represents the vehicle mass, J_{wf} and J_{wr} represent the polar moments of inertia of the front and rear wheels respectively, J_m the equivalent moment of inertia of

the engine at the crankshaft. The inertial effect of the engine is reduced at the driving wheels by multiplying it by the square of the current gear ratio G_R, which is selected as a function of the vehicle forward speed. Finally, the last term of Eq. 5.5 includes the aerodynamic drag. The drag induced by the front tyre lateral forces when the vehicle is cornering is not accounted for. This, though, together with the above approximations causes a negligible loss in accuracy, as the feedback longitudinal control scheme is self correcting.

In vigorous manoeuvring, e.g. when following a reconstructed racing line at racing speed, the longitudinal force F_{vx} may exceed the actual friction forces available from the tyre model to accelerate or brake the vehicle. When this occurs, the vehicle velocity becomes smaller or greater than the target velocity as the tyres can not generate sufficient driving or braking force respectively. In turn, Eq. 5.4 returns even greater values (in magnitude) for the required vehicle forward acceleration, hence causing the longitudinal thrust to increase as well. Eventually, one or more of the vehicle wheels spin or lock. This compromises the vehicle lateral control and the path following simulation will certainly fail. In order to prevent this, a strategy to limit the longitudinal thrust within acceptable values has been devised. The module of the driver model which deals with the longitudinal control takes the individual longitudinal slip of the wheels as inputs. These inputs are used in "anti-spin" and "anti-lock" functions which return 1 if the input is within predetermined values and 0 otherwise, but with a smooth and relatively sharp transition. This is obtained by employing a combination of *sin*(*arctan*()) functions inspired by the Magic Formula tyre model (Bakker, Nyborg and Pacejka, 1987). For the front wheels, to which only braking torque is applied, the expression for the "anti-lock" function reads:

$$A_{LF} = A_{RF} = 0.5 \cdot \sin\left(\arctan\left(B \cdot \left(k + L_{SL}\right)\right)\right) + 0.5 \qquad \text{Eq. 5.6}$$

where k is the wheel longitudinal slip and L_{SL} is the pre-determined maximum longitudinal slip (in magnitude) allowed. For the rear wheels, to which, instead, either driving or braking torque is applied, the combined "anti-spin" and "anti-lock" function reads:

$$\begin{aligned} A_{LR} = A_{RR} = &\left(-0.5 \cdot \sin\left(\arctan\left(B \cdot \left(k - L_{SL}\right)\right)\right) + 0.5\right) \times \\ &\left(0.5 \cdot \sin\left(\arctan\left(B \cdot \left(k + L_{SL}\right)\right)\right) + 0.5\right) \end{aligned} \qquad \text{Eq. 5.7}$$

The constant parameter B in Eqs. 5.6 and 5.7 is used to adjust the slope of the functions during the transitions from 0 to 1 and vice versa. Figure 5.3 shows the output of these functions when the longitudinal slip varies from –20 % to +20 %, with L_{SL} set equal to 10 % and B set equal to 10. The "anti-spin" and "anti-lock" functions for each wheel are used to factor the longitudinal thrust returned by Eq. 5.5, so that the actual longitudinal thrust applied to the vehicle reads:

$$F_{vx} = F_{vx} \times A_{LF} \times A_{RF} \times A_{LR} \times A_{RR} \qquad \text{Eq. 5.8}$$

Ideally, the value for the maximum longitudinal slip L_{SL} should be chosen in such a way that the tyres may operate up to a value for the longitudinal slip which is as close as possible to that giving the maximum longitudinal force. Such value, though, would

depend on many parameters, e.g. wheel slip angles, vertical loads, etc. A constant value for L_{SL} equal to 10 % was chosen to achieve a good compromise.

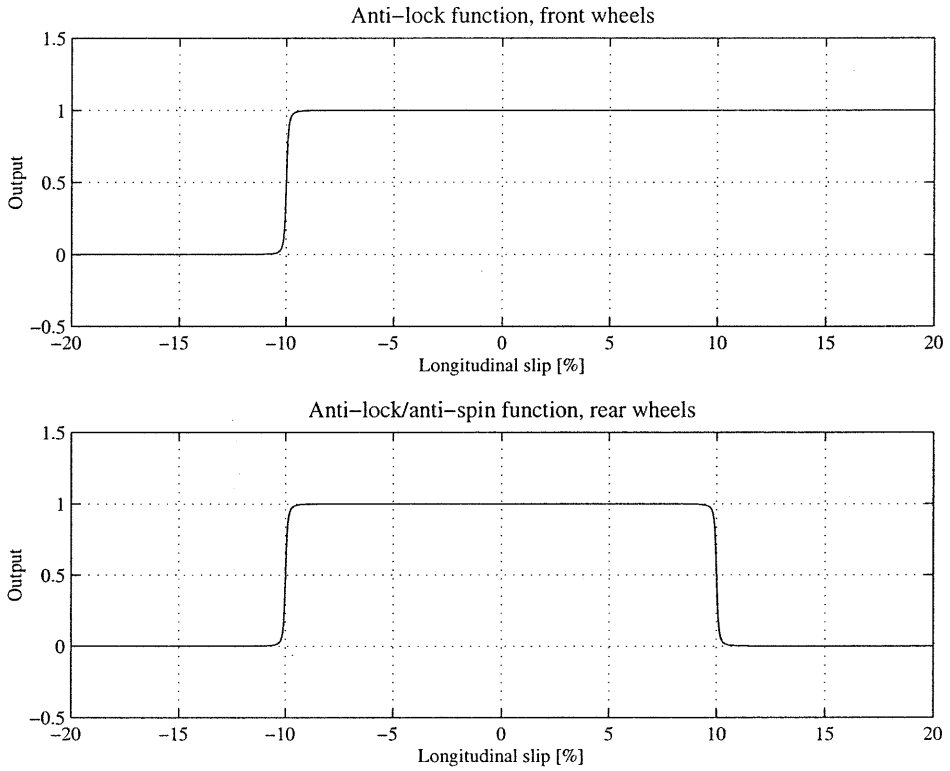


Figure 5.3 "Anti-spin" and "anti-lock" functions.

Having determined the longitudinal thrust, it is now necessary to evaluate the equivalent driving or braking torque which is to be applied at each wheel in order to produce such thrust. In order to do so, we will not consider the actual maximum braking or driving torque available as limiting boundaries. Instead, we will assume that unlimited power is available to drive or brake the vehicle. Only at the end of the path following simulation, during the post-processing phase of the results, will we use the model of the engine output torque and of the braking system in order to evaluate the throttle and brake control input. In certain conditions it will then be possible to obtain values for the throttle and brake input greater than 100 % or smaller than −100 % respectively, meaning that the required velocity profile exceeds the vehicle performance boundary. One thing, though, must be taken into account. In order to maintain the proper balance for the vehicle when braking, the engine braking torque must be accounted for during the simulation.

Consider firstly when F_{vx} is positive (driving). In this case all the torque is applied to the rear wheels. Since in the SAE sign conventions the driving torque is negative (see also figure 5.2), we may write:

$$T_{LR} = T_{RR} = -\frac{1}{2} \cdot F_{vx} \cdot R_r \qquad \text{Eq. 5.9}$$

The torque transfer due to the limited slip differential must then be added. When driving, the input torque to the differential cage reads:

$$T_{in} = -F_{vx} \cdot R_r \qquad \text{Eq. 5.10}$$

Then, the actual torque transfer may be evaluated using the first of Eq. 5.1 and Eq. 5.2. Recalling the SAE sign convention once more, the wheel rotational velocity is negative when the vehicle proceeds forward. Therefore, Eq. 5.2 returns a negative value for the torque transfer when the rear left wheel spins faster than the rear right wheel. Hence the extra driving torque must be added to the slower wheel, i.e. the right rear:

$$\begin{aligned} T_{LR} &= T_{LR} - \Delta T \\ T_{RR} &= T_{RR} + \Delta T \end{aligned} \qquad \text{Eq. 5.11}$$

Finally, for the front wheels the torque applied will simply be equal to zero:

$$T_{LF} = T_{RF} = 0 \qquad \text{Eq. 5.12}$$

Consider now when F_{vx} is negative (braking). In this case the braking torque available from the engine must be taken into account in order to maintain the proper balance for the vehicle under braking. Using the mean value of the rear wheel rotational velocities, the engine rotational speed in revolutions per minute is evaluated as follows (note that such rotational velocity must be positive):

$$rpm = -\frac{15 \cdot (\omega_{LR} + \omega_{RR}) \cdot G_R}{\pi} \qquad \text{Eq. 5.13}$$

The maximum engine braking torque available E_{brk} is then evaluated from the experimental engine map using the engine rotational speed and considering the throttle input equal to 0. Two situations may now arise. If the negative longitudinal thrust is less than the engine braking effect:

$$|F_{vx}| \leq \left| \frac{E_{brk} \cdot G_R}{R_r} \right| \qquad \text{Eq. 5.14}$$

the braking torque is only applied to the rear axle. In this case Eqs. 5.9 to 5.12 still hold, with the only difference that the limited slip differential torque transfer must now be evaluated using the second of Eq. 5.1. If, instead, the longitudinal thrust exceeds the engine braking effect, the residual negative force F_{res} must be applied through the brakes. The expression for the residual force reads:

$$F_{res} = F_{vx} + \frac{E_{brk} \cdot G_R}{R_r} \qquad \text{Eq. 5.15}$$

The sign "+" in Eq. 5.15 is dictated by the convention that the engine braking torque is positive. The constant parameters B_f and B_r determine the fixed ratio of the braking torque applied to the front and rear axles respectively. Taking also into account the different front and rear wheel radii, the individual braking torque applied to each wheel

reads:

$$T_{LF} = T_{RF} = -F_{res} \cdot \frac{R_f \cdot R_r \cdot B_f}{R_r \cdot B_f + R_f \cdot B_r}$$
$$T_{LR} = T_{RR} = E_{brk} \cdot G_R - F_{res} \cdot \frac{R_f \cdot R_r \cdot B_r}{R_r \cdot B_f + R_f \cdot B_r} \qquad \text{Eq. 5.16}$$

Finally, the effect of the limited slip differential must be added to the torque applied to the rear wheels. The input torque to the differential cage reads:

$$T_{in} = E_{brk} \cdot G_R \qquad \text{Eq. 5.17}$$

Then, Eqs. 5.1, 5.2 and 5.11 are used as above.

The value for the torque applied to each wheel may be entered in the vehicle model directly and the path following simulation may be performed. After the simulation, it is interesting to derive the longitudinal control history in terms of throttle/brake input u_{tb}. When the vehicle is braking, the torque applied to the front wheels allows to identify the brake control input easily. As far as the vehicle model is conceived, the brake control input simply factors the maximum braking torque available. Hence, the control input corresponding to the front wheel torque T_{LF} and T_{RF} reads:

$$u_{tb} = -\frac{(T_{LF} + T_{RF})}{B_{T\_MAX} \cdot B_f} \qquad \text{Eq. 5.18}$$

where B_{T_MAX} is the maximum torque available from the braking system.

When no torque is applied to the front wheels, it means that all the driving/braking torque is supplied by the engine and applied to the rear wheels. In this case the throttle input may be identified using the model of the engine output torque, i.e. the experimental engine map. The engine map is a two-dimensional look-up table which yields the engine output torque E_{trq} as a function of the throttle input and the engine rotational velocity. In the present case, since the required engine output is known, the task is to determine the input which yields such torque output. Figure 5.4 shows the procedure graphically. Firstly, the engine rotational velocity must be evaluated using the mean rotational velocity of the rear wheels, see Eq. 5.13. Then, the column of the engine map, representing the engine output torque at fixed rotational velocity (in *rpm*) and variable throttle position is identified. This may be accomplished by doing a one-dimensional linear interpolation for each of the rows of the engine map. Finally, the throttle input u_{tb} corresponding to the actual torque applied to the rear wheels (T_{LR} + T_{RR}) is identified by doing one more linear interpolation. When driving, if the engine output torque is less than the actual torque applied to the wheels, the algorithm will extrapolate a value for the throttle input which will be greater than 100 %.

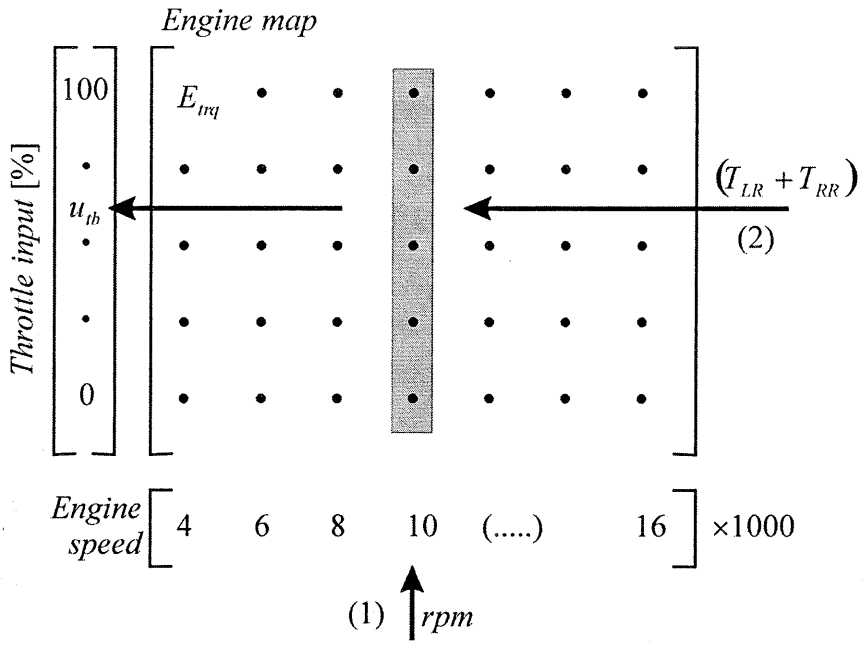


Figure 5.4 Identification of the throttle control input.

5.4. RESULTS

In this section various path following tasks of increasing difficulty are set up in order to prove the effectiveness of the driver model. Firstly, a short double lane change manoeuvre has been designed specifically to simulate moderate "*g*" manoeuvres. Then, the racing lines obtained using the algorithm described in chapter 3 have been used to simulate a lap of the Catalunya circuit in Barcelona and a lap of the Japanese circuit of Suzuka. Finally, the vehicle configuration has been altered in order to make the vehicle either strongly understeer or oversteer. The simulation of the lap of the Catalunya circuit has been repeated for both cases. The vehicle parameters and the tyre model used are representative of a Formula One car in race trim configuration according to the 1999 technical specifications[2]. Further details are given in Appendix A.

Each path following task is documented with a set of six figures, whose contents are outlined here:

- The first figure shows the vehicle trajectory compared with the ideal path to follow;
- The second figure shows firstly the vehicle lateral off-set, which allows to quantify the path tracking error in terms of the distance of the vehicle mass centre from the ideal path. A second graph shows the vehicle longitudinal velocity compared with the target velocity profile. This figure with the first one are intended to investigate the accuracy of the solution to the path following task;
- The third figure shows the vehicle lateral and longitudinal controls, i.e. the steer angle and the throttle/brake control inputs. The simulated steer angle is also compared with the steer angle measured on the real car (except for the double lane change manoeuvre for which such data were obviously not available) and this is

[2] Vehicle parameters and tyre data supplied are courtesy of Benetton Formula Ltd.

indicative of both the quality of the driver model as well as of the vehicle model;

- The fourth figure shows the vehicle yaw rate, the vehicle lateral velocity and the difference between the front axle and rear axle slip angles. This information gives a clear indication of the vehicle attitude through corners and is useful to compare different vehicle set-ups;
- Finally, the last two figures show the tyre utilisation indexes for each individual wheel. The tyre utilisation index is defined rigorously in Appendix A. It may be described as the percent of lateral and longitudinal force which is used from one tyre in combined slip conditions compared to the maximum force available. These graphs indicate how close to its performance limit the vehicle is operating.

5.4.1. Moderate "g" manoeuvres: the double lane change

An ideal path representing a double lane change manoeuvre has been generated by defining a curvature function varying between 0 and 0.04, corresponding to a minimum cornering radius of 25 meters, see figure 5.5. The intended vehicle longitudinal velocity has been set equal to a constant value of 16 meters per second. This implies a lateral acceleration at the apex of the turn of about 1.05 g, which is not very demanding for a Formula One car.

Figure 5.5 shows the vehicle trajectory superimposed on the ideal path. The two lines match almost perfectly and only the first graph in figure 5.6 reveals a maximum path tracking error of about 0.15 m. The vehicle model completes the manoeuvre maintaining the ideal velocity as the second plot in figure 5.6 shows. Only when the vehicle is cornering is there a small error as a result of the drag induced by the steering of the road wheels. The throttle input also increases slightly during cornering, to partially compensate for the difference between the vehicle actual velocity and the target velocity, see the second graph in figure 5.7.

The steer angle, together with the vehicle yaw rate and lateral velocity show how the vehicle follows the ideal path with a smooth manoeuvre, without any oscillatory pattern induced by the coupling between driver model and vehicle model. Finally, figures 5.9 and 5.10 show that the level of tyre lateral saturation reaches at most the value of 60 – 65 %, proving that the manoeuvre is well within the vehicle performance limits.

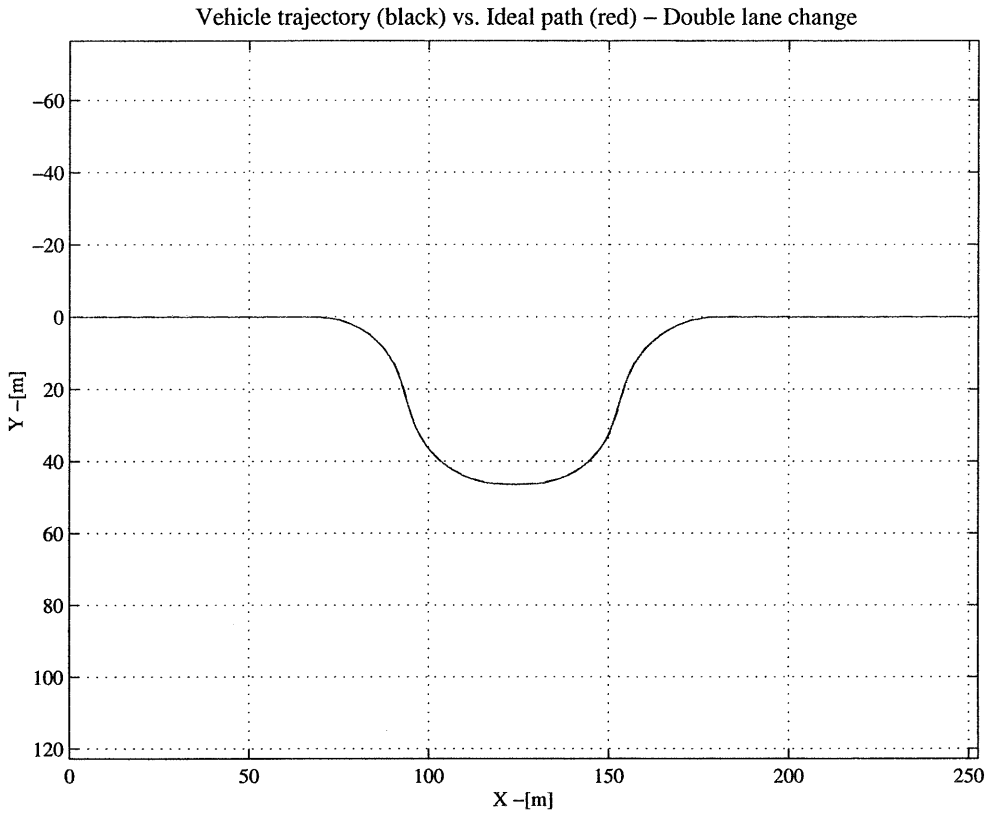


Figure 5.5 Double lane change, vehicle trajectory compared with the ideal path.

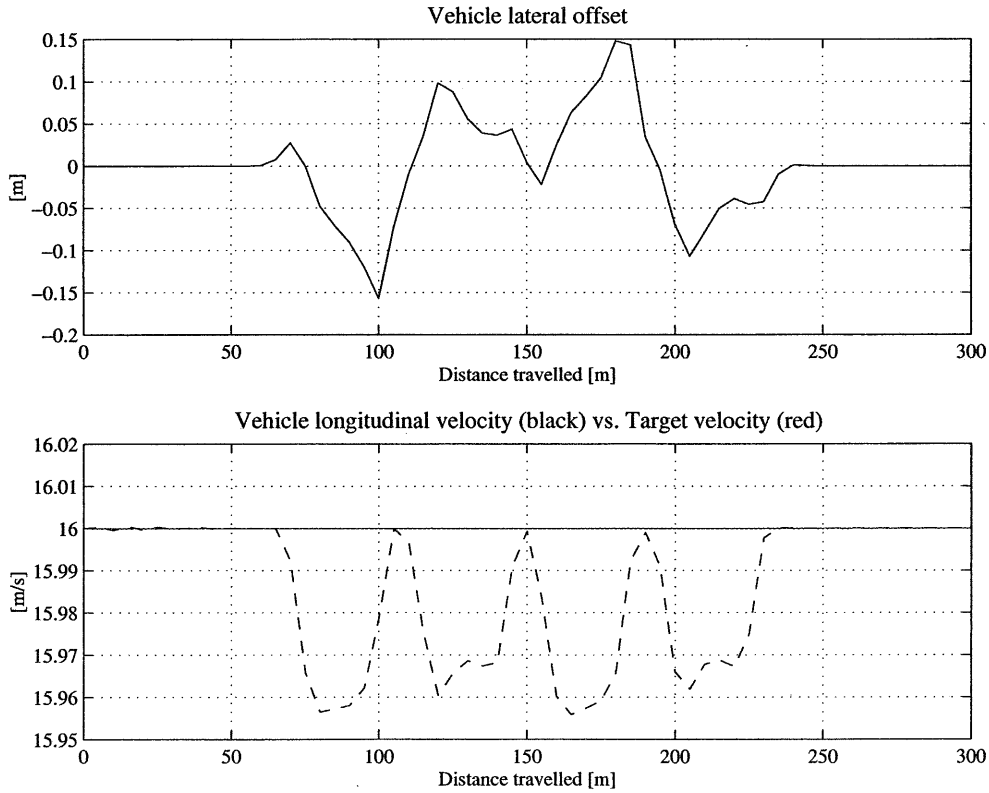


Figure 5.6 Path tracking error and vehicle longitudinal velocity.

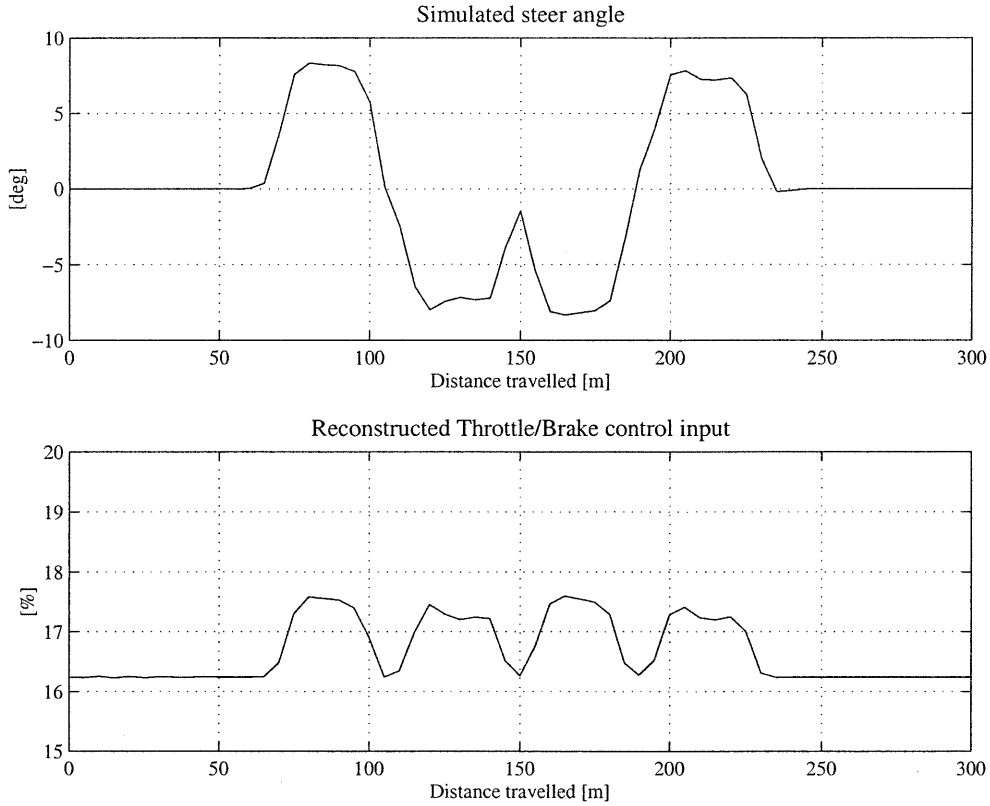


Figure 5.7 Vehicle lateral and longitudinal controls.

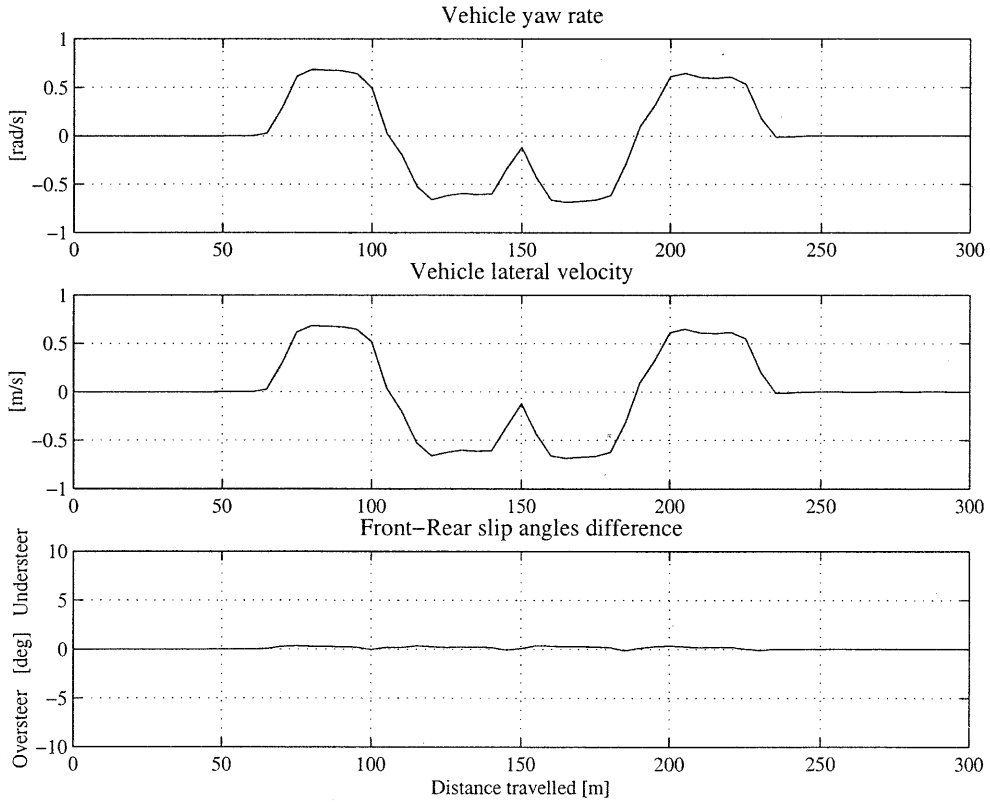


Figure 5.8 Vehicle yaw rate, lateral velocity and attitude.

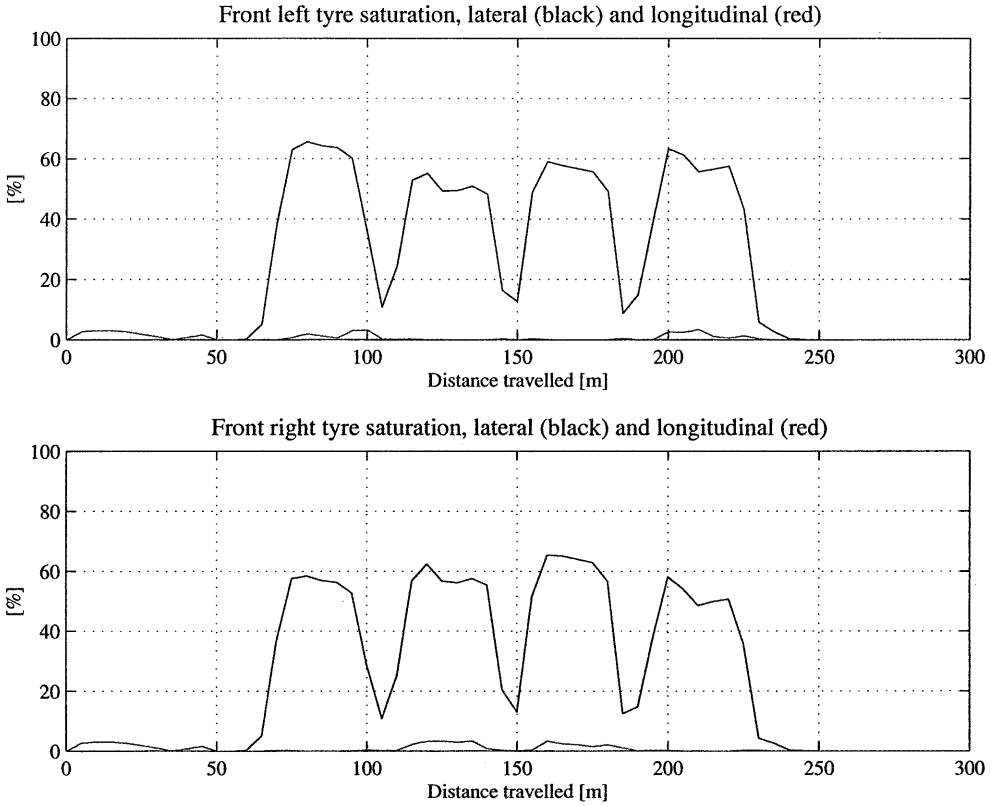


Figure 5.9 Front tyres lateral and longitudinal utilisation indexes.

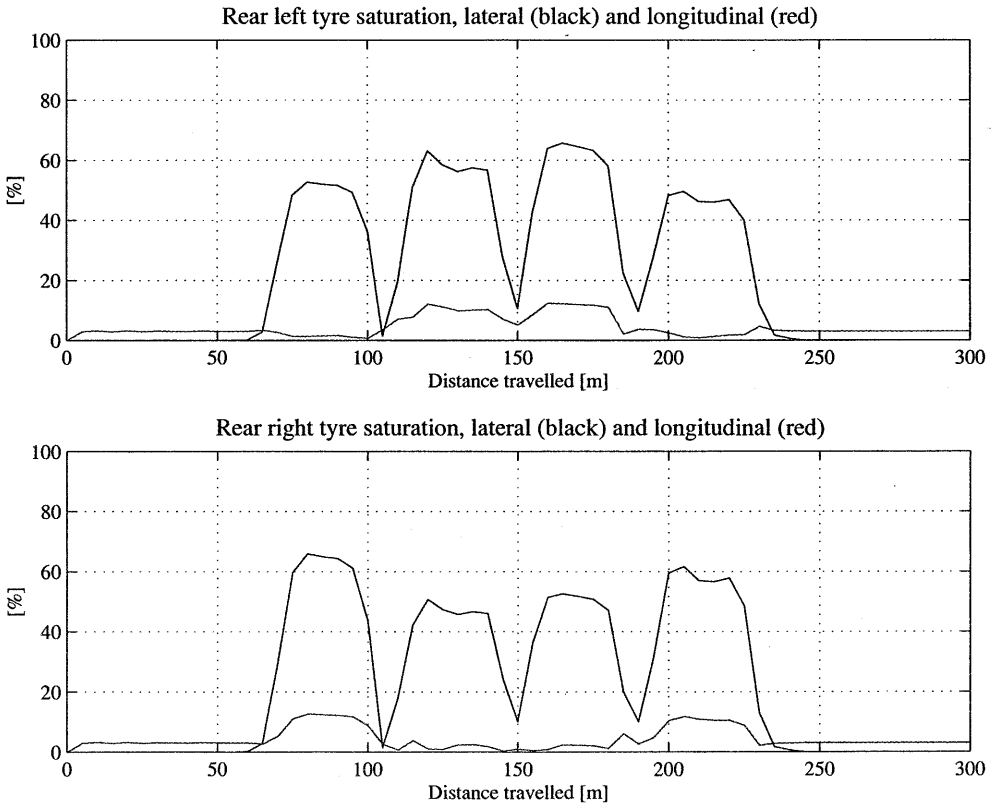


Figure 5.10 Rear tyres lateral and longitudinal utilisation indexes.

5.4.2. High "g" manoeuvres: lap simulations

Using the method described in chapter 3, the racing lines for two laps from two different circuits have been reconstructed. Then, the driver model has been asked to follow these paths at racing speed. The two sets of results are presented here. The first refers to a lap of the Catalunya circuit in Barcelona (figures 5.11 to 5.16) and the second to a lap of the Suzuka circuit (figures 5.17 to 5.22). Realistic vehicle parameters for the car in race trim configuration were used. The simulations, though, could not be performed using the tyre data as they were given, since the tyre forces were too low. This happens commonly, since the laboratory conditions for the tyre tests will never be the same as the track conditions. Racing teams usually compare experimental data with simulation results, e.g. measured *g-g* diagrams are compared with simulated steady-state performance envelopes, in order to find the proper scaling of the tyre forces which yields good agreement between simulations and measurements. This procedure was not possible here, since sufficient data were not available. Thus, the scaling coefficient was found by progressively increasing the tyre forces until the simulation of the lap could be accomplished with sufficient robustness. In the end, the tyre forces have been increased by 15 %.

For the lap of the Catalunya circuit, Figure 5.11 together with the first plot of figure 5.12, show the excellent path tracking capability of the driver model, with the lateral off-set of the vehicle centre of mass never exceeding 0.15 m. Also the target velocity is matched very well throughout the whole lap, as the second plot of figure 5.12 shows. The steer angle returned by the driver model (figure 5.13) is in good agreement with that measured on board the car. The reconstruction of the throttle/brake control input yields as well reasonable results, with the maximum throttle input close to 100 % which indicates the accuracy of the vehicle model in this respect. However, the signal is very oscillatory. This is a consequence of irregularities in the target velocity profile, which is obtained by filtering the velocity recorded on the real car (see §3.2). Such irregularities affect the vehicle forward acceleration evaluated using Eq. 5.4 and determine the noise in the longitudinal control. Figure 5.14 shows how the vehicle follows the intended path smoothly, without oscillation and with a mild tendency to understeer. This is finally confirmed also by the graphs of the tyre utilisation indexes, which show that the front tyres are often close to the maximum lateral saturation, while the rear tyres maintain some spare capacity.

The results related to the lap of the Suzuka circuit are in line with those described above, with the exception that the vehicle shows a stronger tendency to understeer (see figure 5.20). The path tracking error is very low for most of the lap, with a maximum values of 0.6 m in correspondence of two corners at $s \cong 800$ m and $s \cong 3800$ m respectively. At the same locations, also the steer angle returned by the driver model is significantly greater than that measured, indicating that probably the required manoeuvre is slightly beyond the performance envelope of the vehicle model. The fact that the path following simulation yields very good results for an entire lap with the exception of few localised sections of the trajectory is in general related to the errors in the reconstruction of the racing line, where some corners may result with a greater curvature than they have in reality.

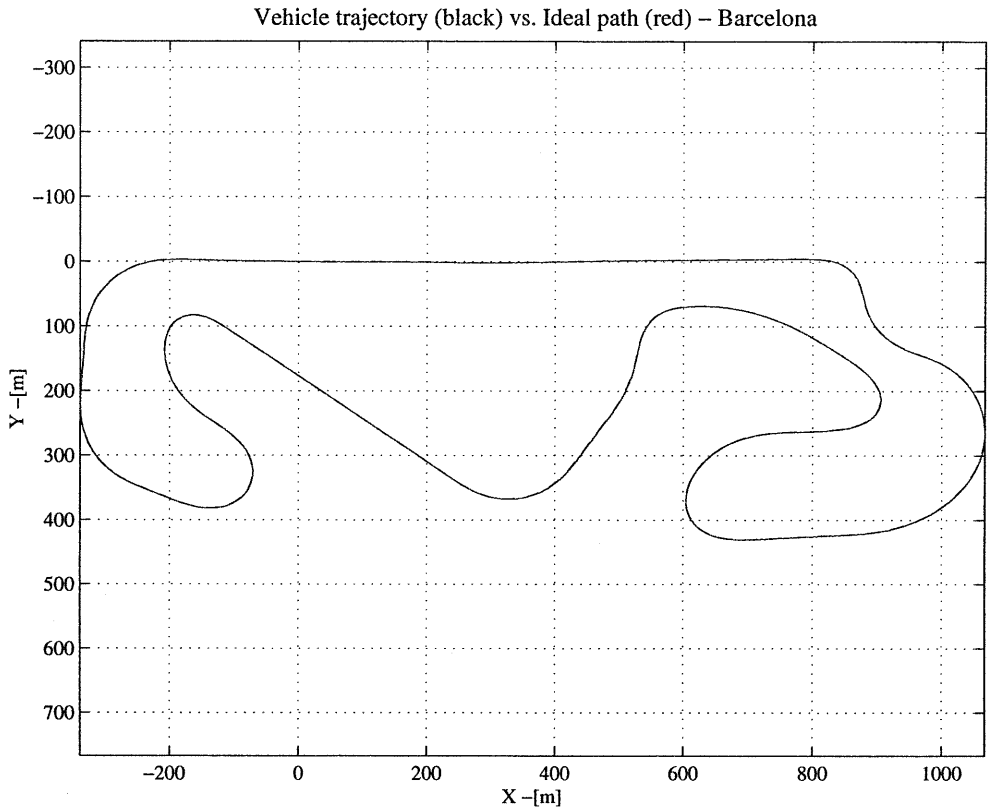


Figure 5.11 Barcelona, vehicle trajectory compared with the ideal path.

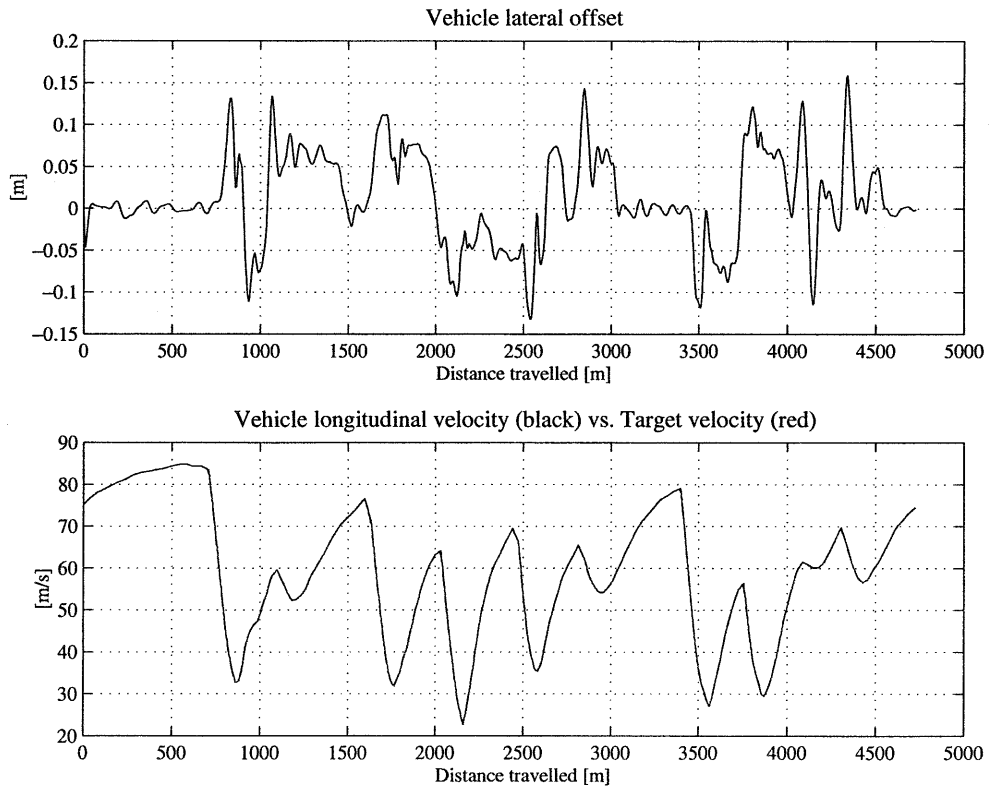


Figure 5.12 Path tracking error and vehicle longitudinal velocity.

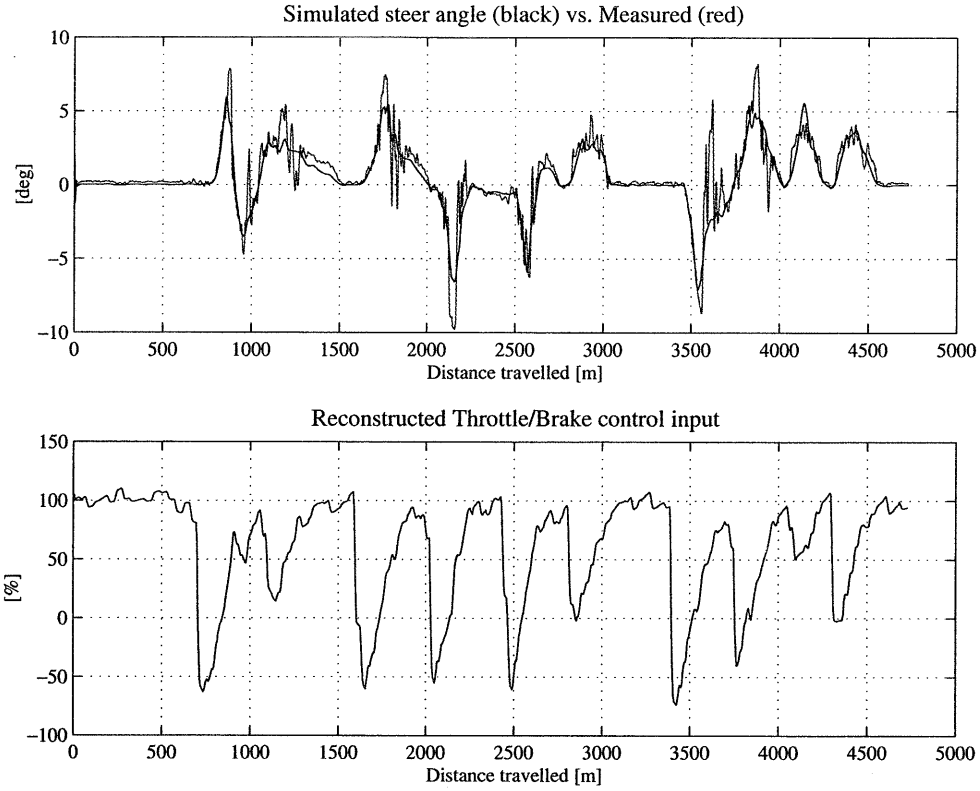


Figure 5.13 Vehicle lateral and longitudinal controls.

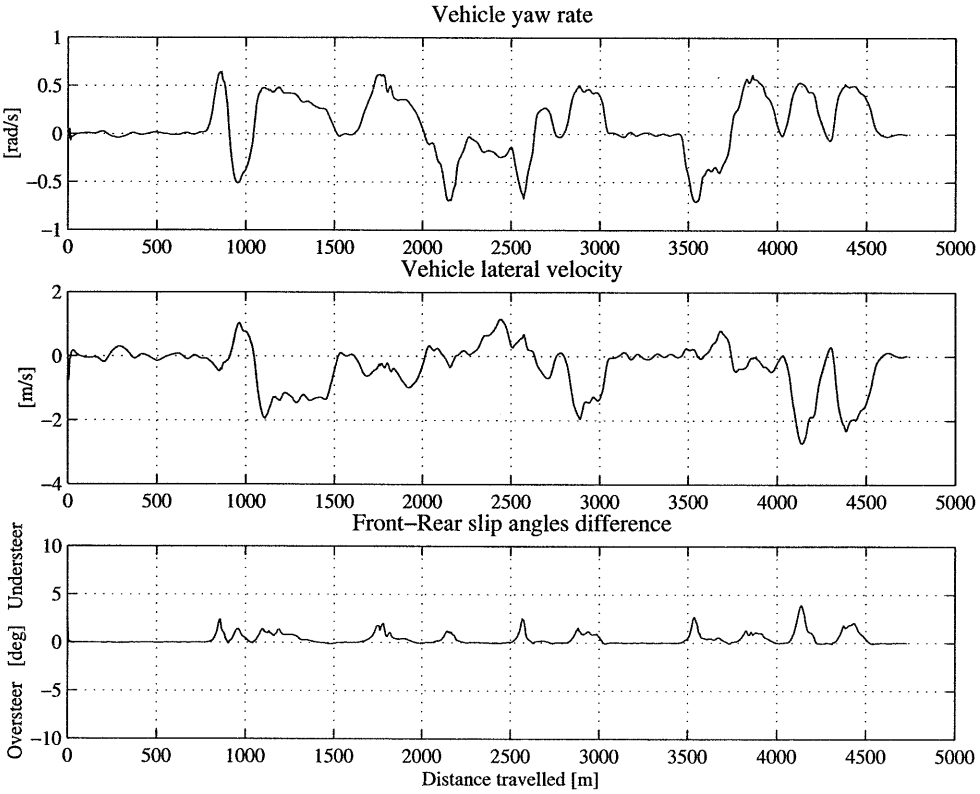


Figure 5.14 Vehicle yaw rate, lateral velocity and attitude.

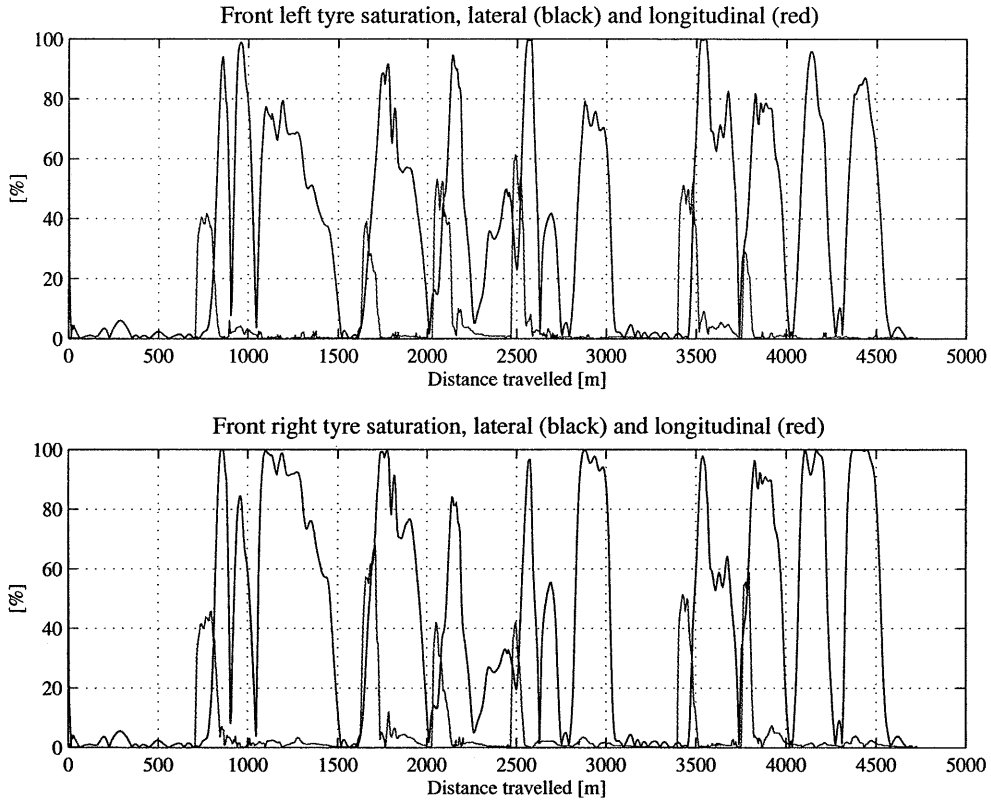


Figure 5.15 Front tyres lateral and longitudinal utilisation indexes.

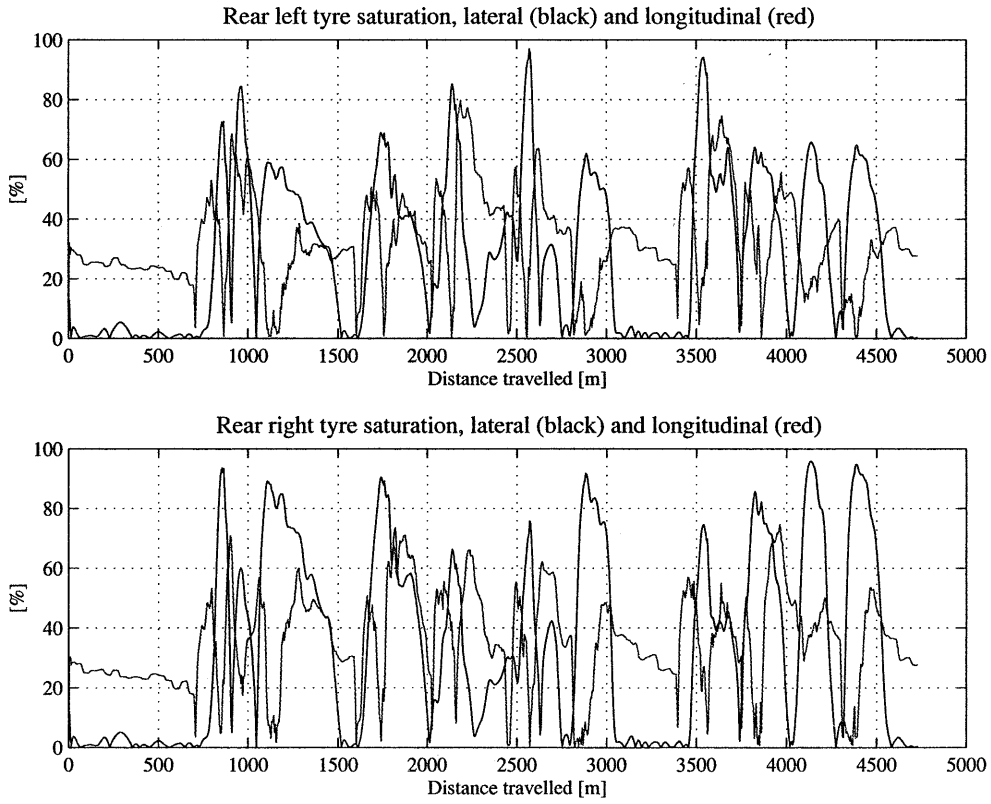


Figure 5.16 Rear tyres lateral and longitudinal utilisation indexes.

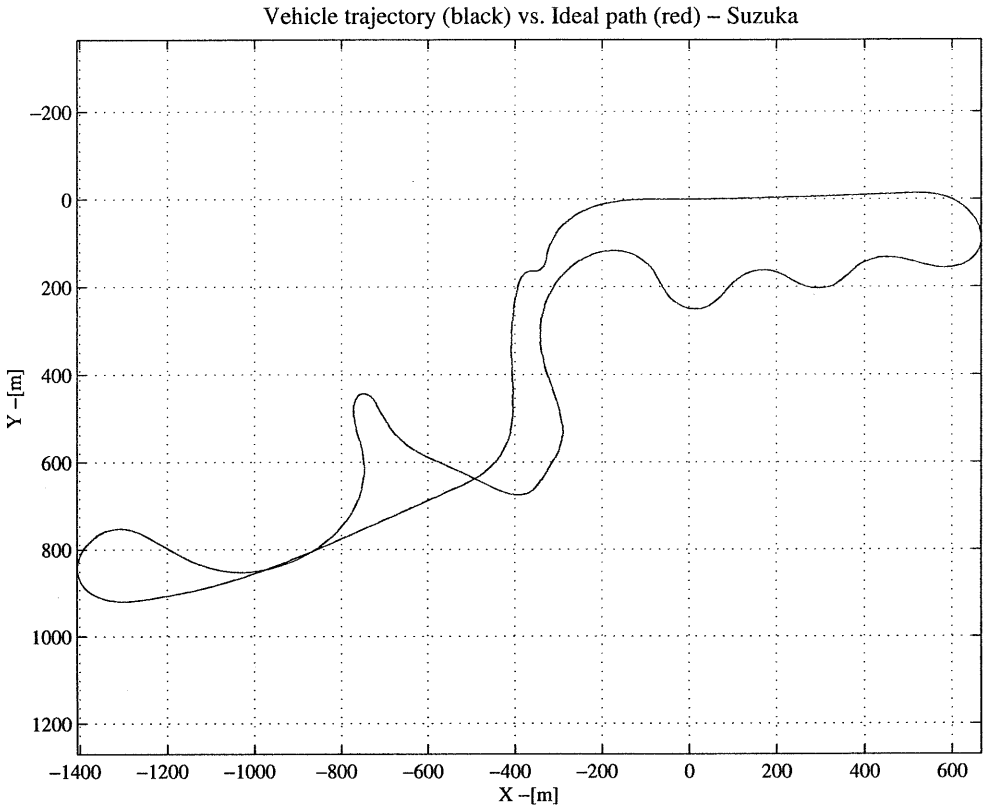


Figure 5.17 Suzuka, vehicle trajectory compared with the ideal path.

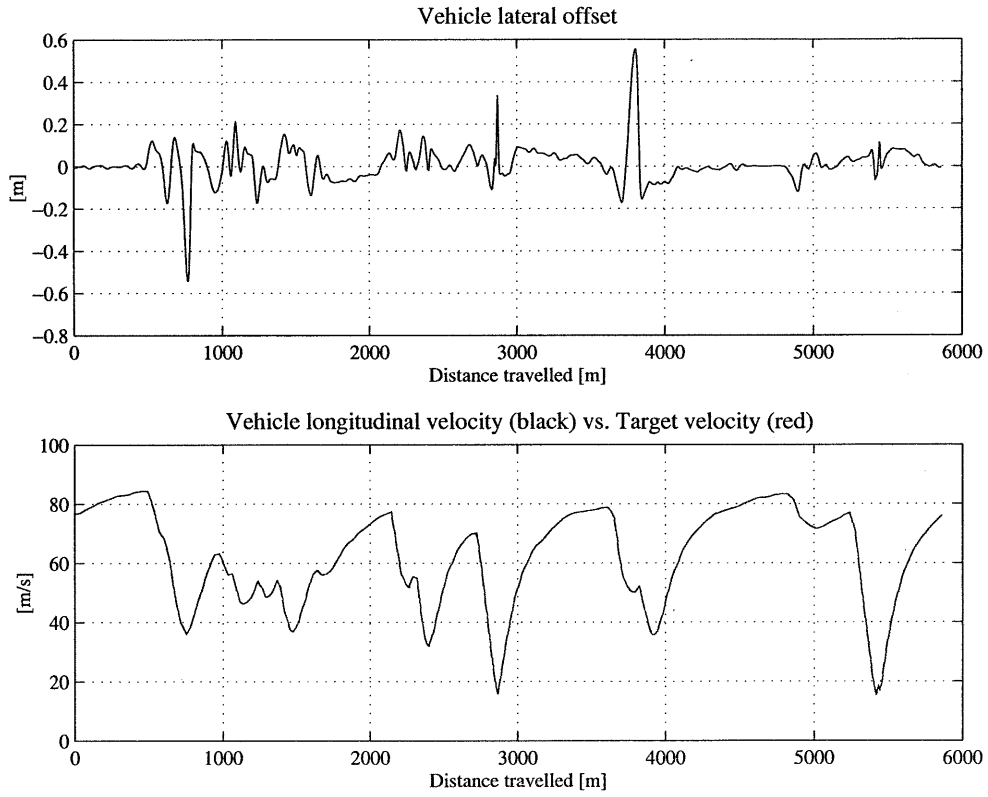


Figure 5.18 Path tracking error and vehicle longitudinal velocity.

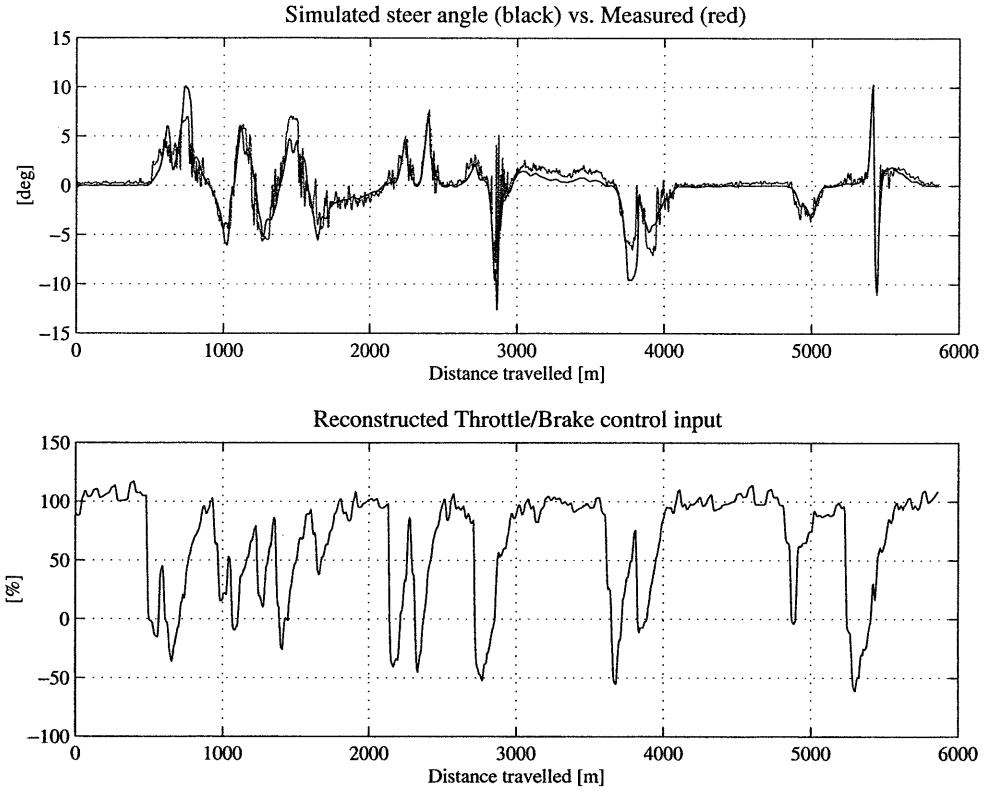


Figure 5.19 Vehicle lateral and longitudinal controls.

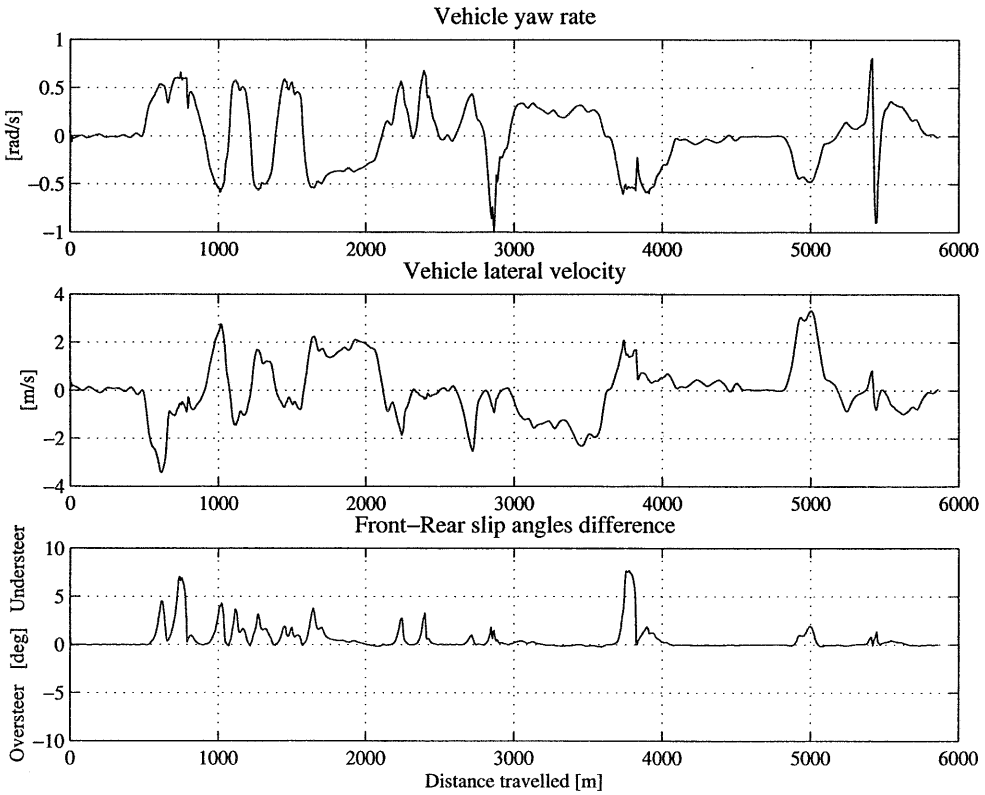


Figure 5.20 Vehicle yaw rate, lateral velocity and attitude.

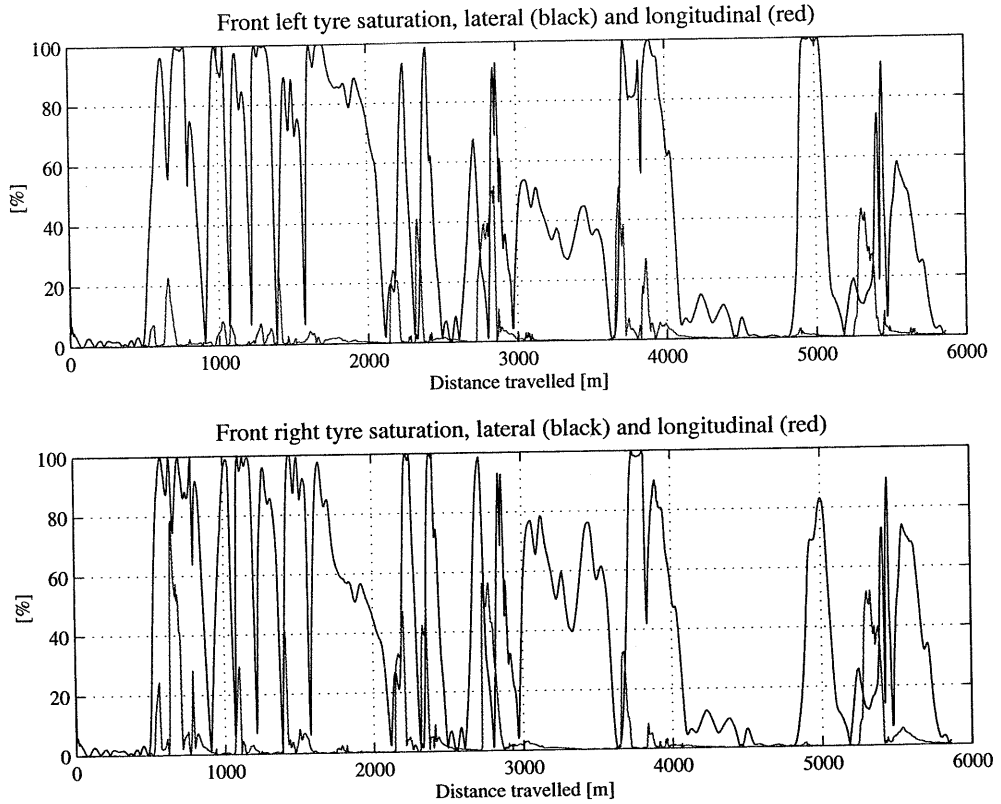


Figure 5.21 Front tyres lateral and longitudinal utilisation indexes.

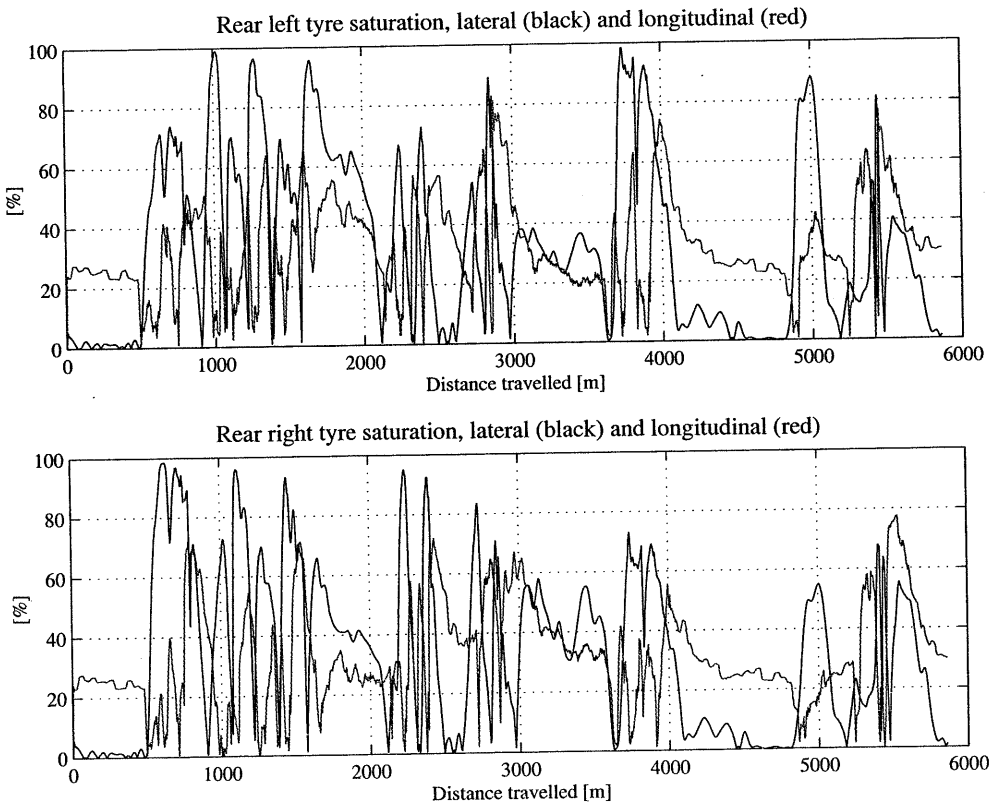


Figure 5.22 Rear tyres lateral and longitudinal utilisation indexes.

5.4.3. Limit manoeuvres: understeer and oversteer vehicles

In order to show the robustness of the driver model in extreme conditions, the vehicle model parameters have been altered in order to change its limit behaviour. The vehicle parameters which influence the load distribution among the tyres have been changed in order to make the vehicle very unbalanced, firstly towards understeer behaviour, then towards oversteer behaviour. These parameters are the roll stiffness distribution and the aerodynamic downforce distribution between the front and rear axles. All the other parameters have remained unchanged, including the scaling coefficient for the tyre forces. The simulation of the lap of the Catalunya circuit has been repeated for both vehicle configurations. Table 5.2 summarises the changes to the vehicle set-up. Clearly, the two configurations adopted here are not realistic, but they are suitable to prove the effectiveness of the driver model in such extreme situations.

VEHICLE SET-UPS			
	Barcelona 1999 (datum)	Understeer case	Oversteer case
Front downforce	41 %	25 %	55 %
Rear downforce	59 %	75 %	45 %
Front roll stiffness	58 %	65 %	50 %
Rear roll stiffness	42 %	35 %	50 %

Table 5.2 Vehicle set-ups for datum, understeer and oversteer vehicle behaviour.

Figures 5.23 to 5.28 refer to the understeer vehicle. The results show a large path tracking error in most of the corners. Even though the vehicle trajectory departs several meters from the ideal path, the steer angle is limited to reasonable values thanks to the saturation functions. In all the cases, even after large excursions from the intended trajectory, the driver model is capable of regaining the original course without any oscillation. As a consequence of the very large slip angles at which the front tyres operate, the induced drag in this case is significant and is responsible for the discrepancies between the vehicle longitudinal velocity and the target velocity. Furthermore, while the vehicle is trying to regain the intended path, the imposed velocity profile requires the vehicle to accelerate while, instead, it is still cornering, as a result of the large off-path excursions. The discrepancy between the actual manoeuvre and the intended manoeuvre is then the cause of the peaks in the reconstructed throttle input, which significantly exceed the value of 100 %, see figure 5.25. Finally, figures 5.27 and 5.28 show the extent of the difference between the front and rear tyre utilisation. While the front tyres generate the maximum lateral force for most of the lap, the rear tyres are only working at up to 60 to 70 % of their capability.

Figures 5.29 to 5.34 refer to the oversteer vehicle. The path tracking error is very small in this case, with the only exception of the penultimate corner, at $s \cong 4200$. Here, the vehicle slides decisively, as may be seen from the second and third plots in figure 5.32. The driver model, though, is able to maintain control by applying opposite steer angle control, see figure 5.31. The third graph in figure 5.32 shows the side-slipping attitude of the vehicle throughout the whole lap, with the rear tyres of the vehicle working most of the time with a larger slip angle than the front tyres. Finally, figures 5.33 and 5.34 show how in this case the rear tyres are on their lateral limit while the front tyre have a marginal amount of spare capacity.

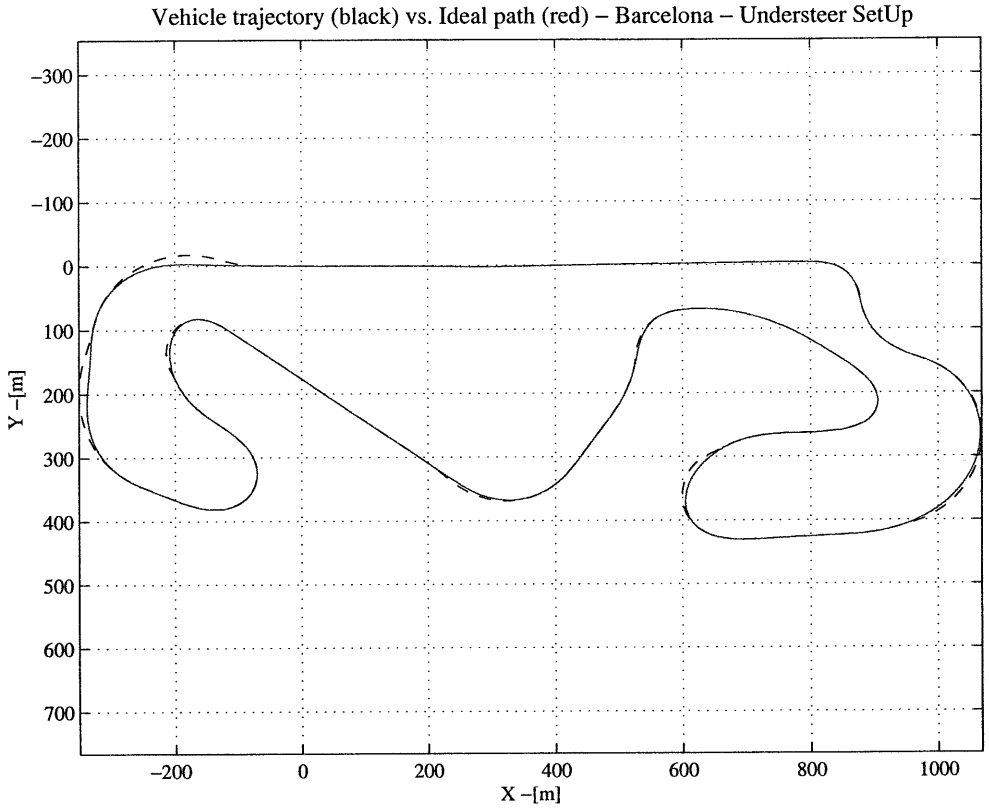


Figure 5.23 Barcelona, understeer vehicle, vehicle trajectory compared with the ideal path.

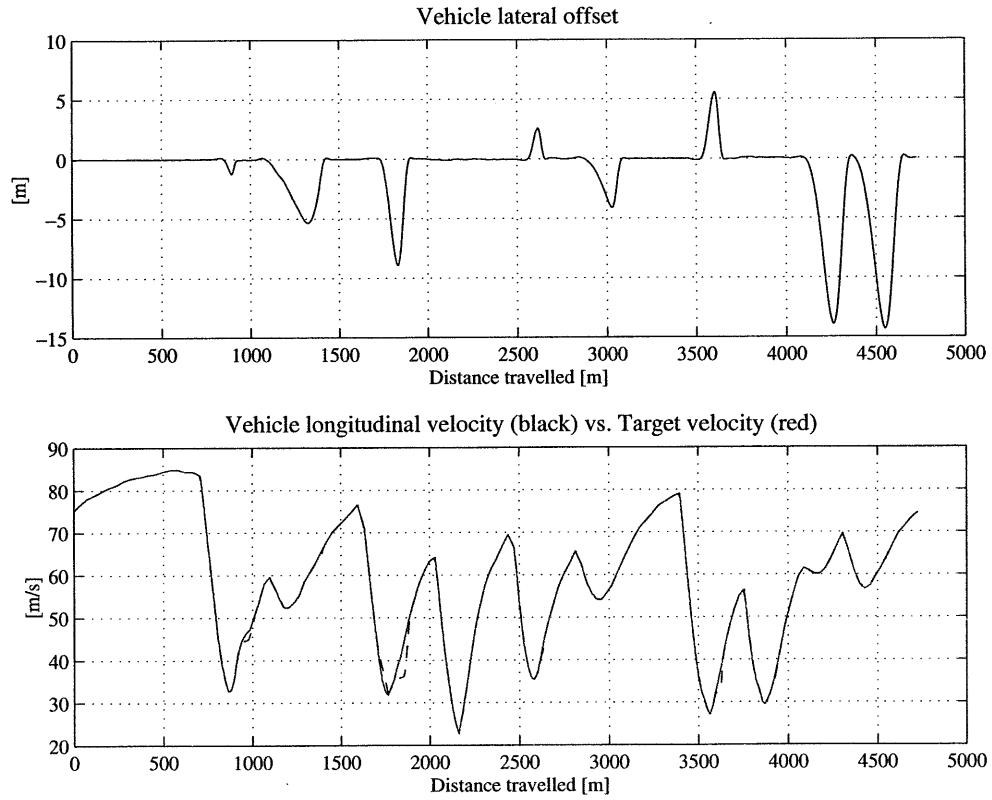


Figure 5.24 Path tracking error and vehicle longitudinal velocity.

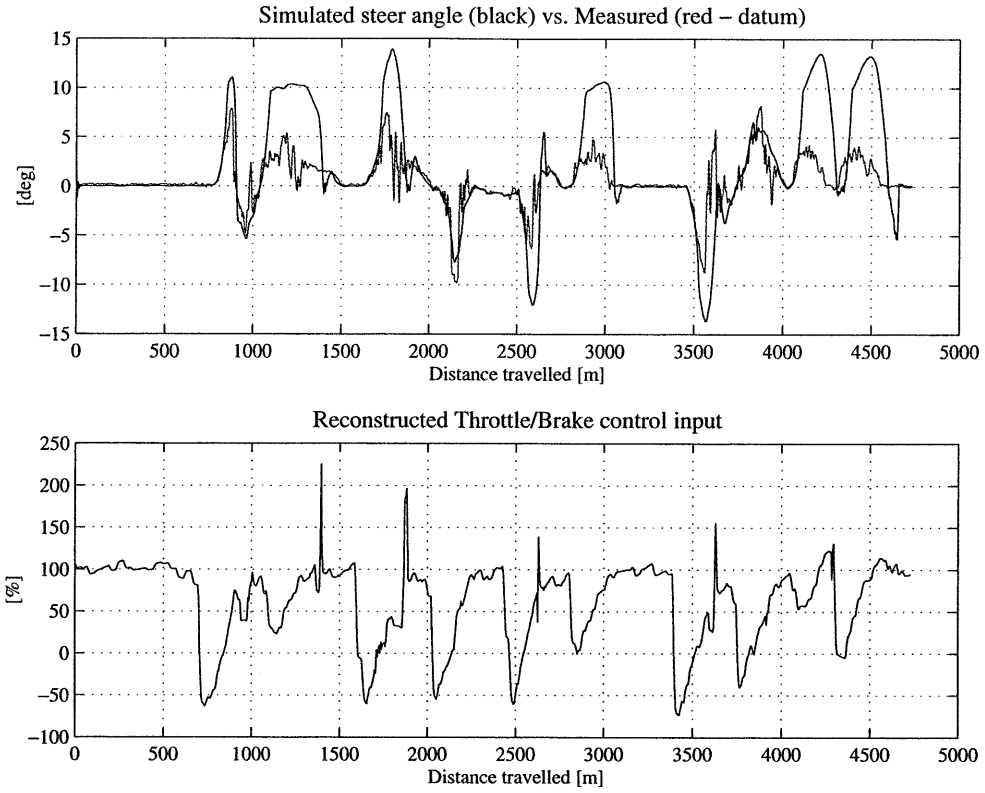


Figure 5.25 Vehicle lateral and longitudinal controls.

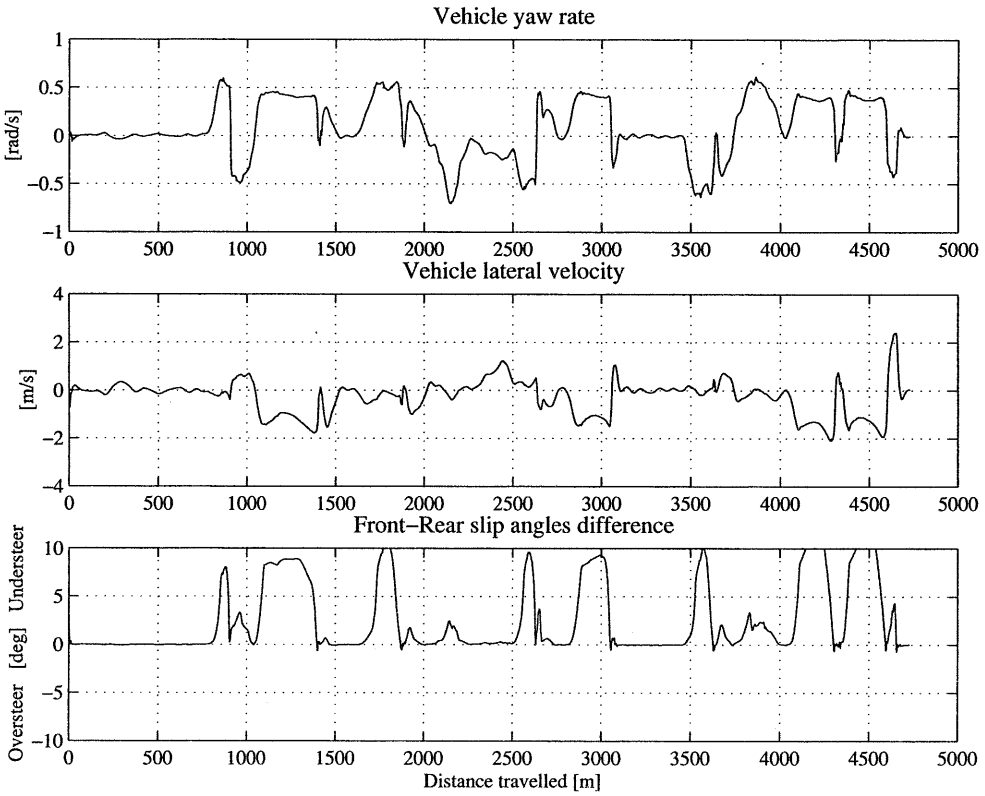


Figure 5.26 Vehicle yaw rate, lateral velocity and attitude.

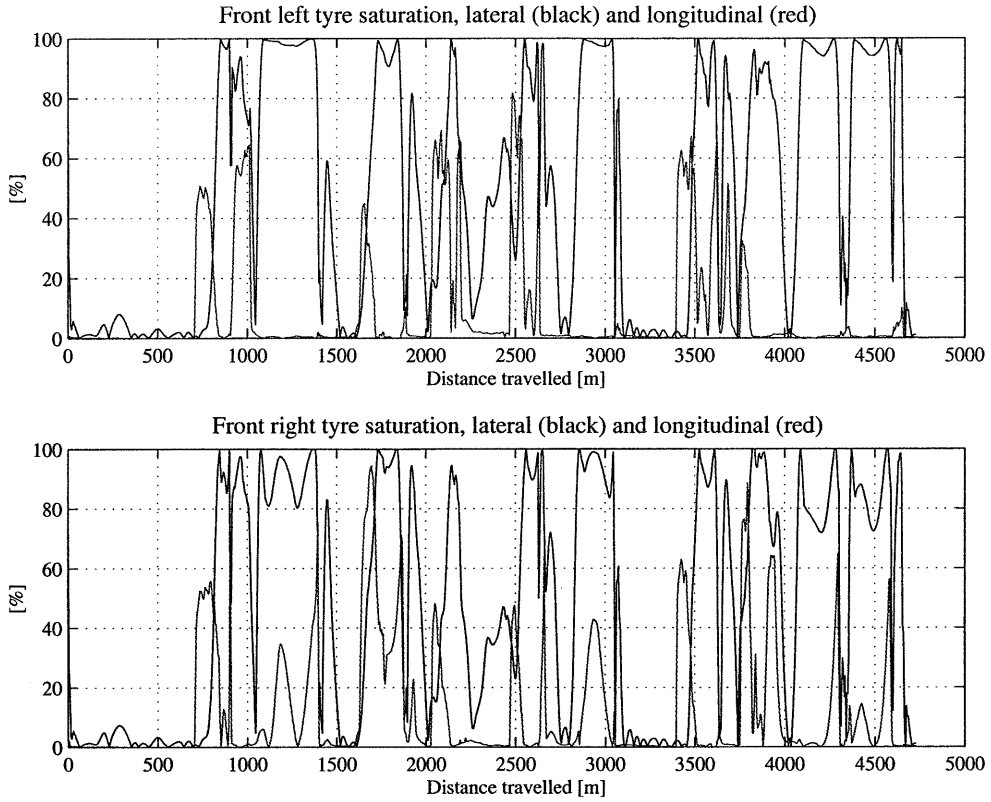


Figure 5.27 Front tyres lateral and longitudinal utilisation indexes.

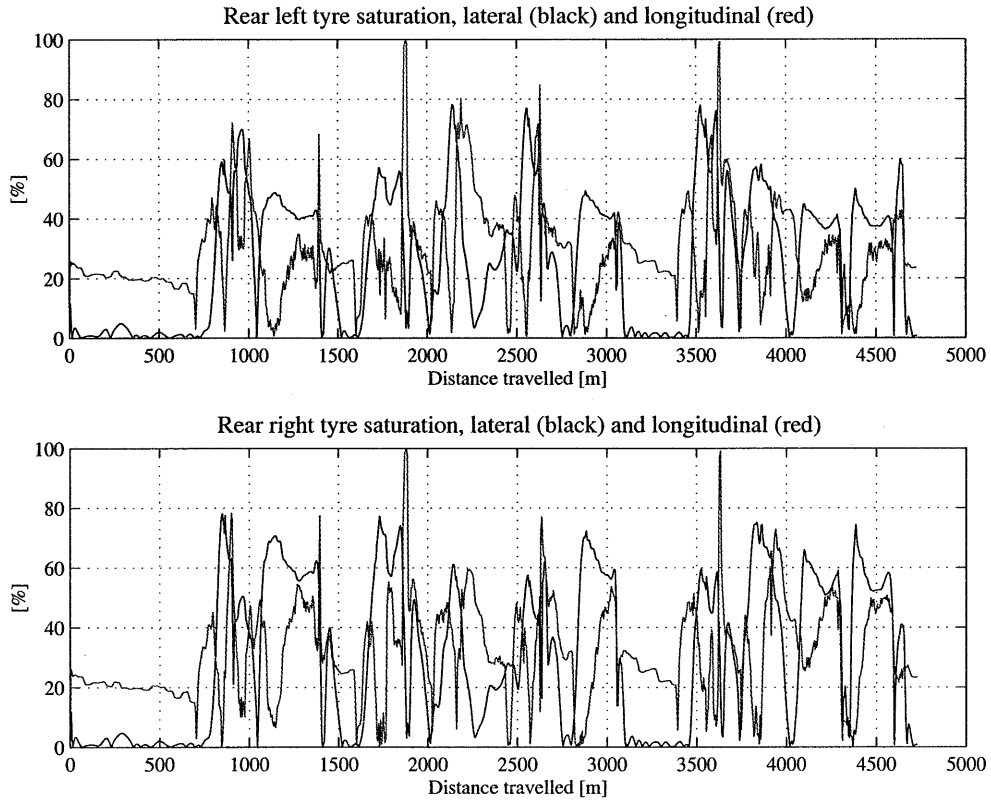


Figure 5.28 Rear tyres lateral and longitudinal utilisation indexes.

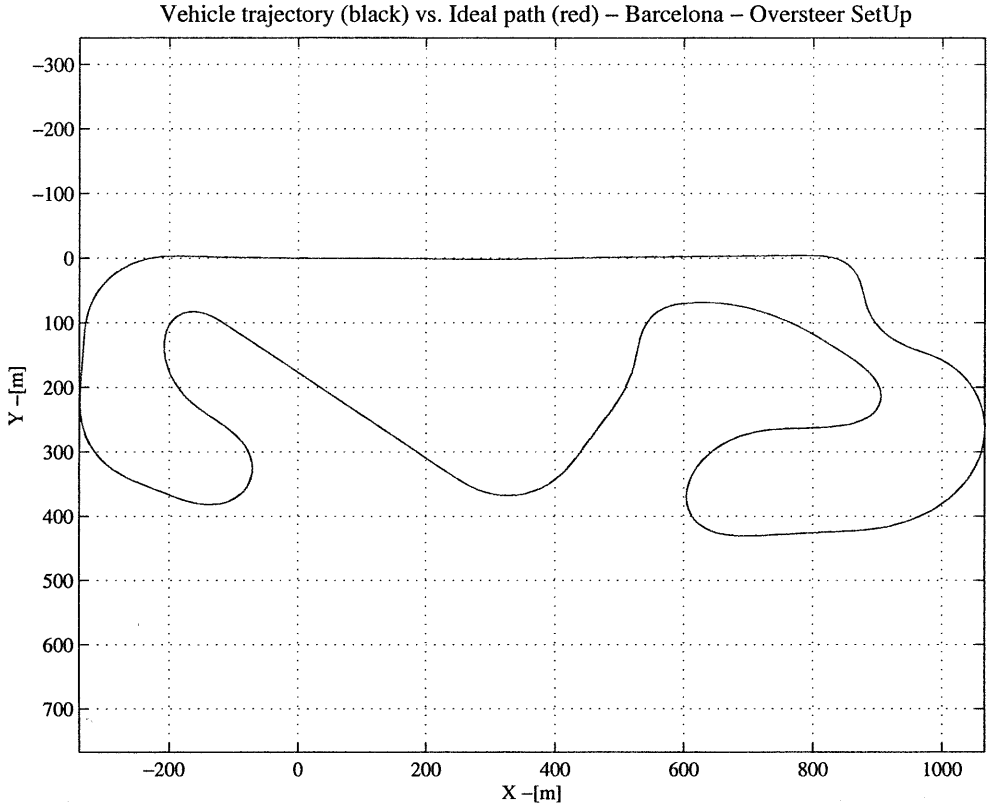


Figure 5.29 Barcelona, oversteer vehicle, vehicle trajectory compared with the ideal path.

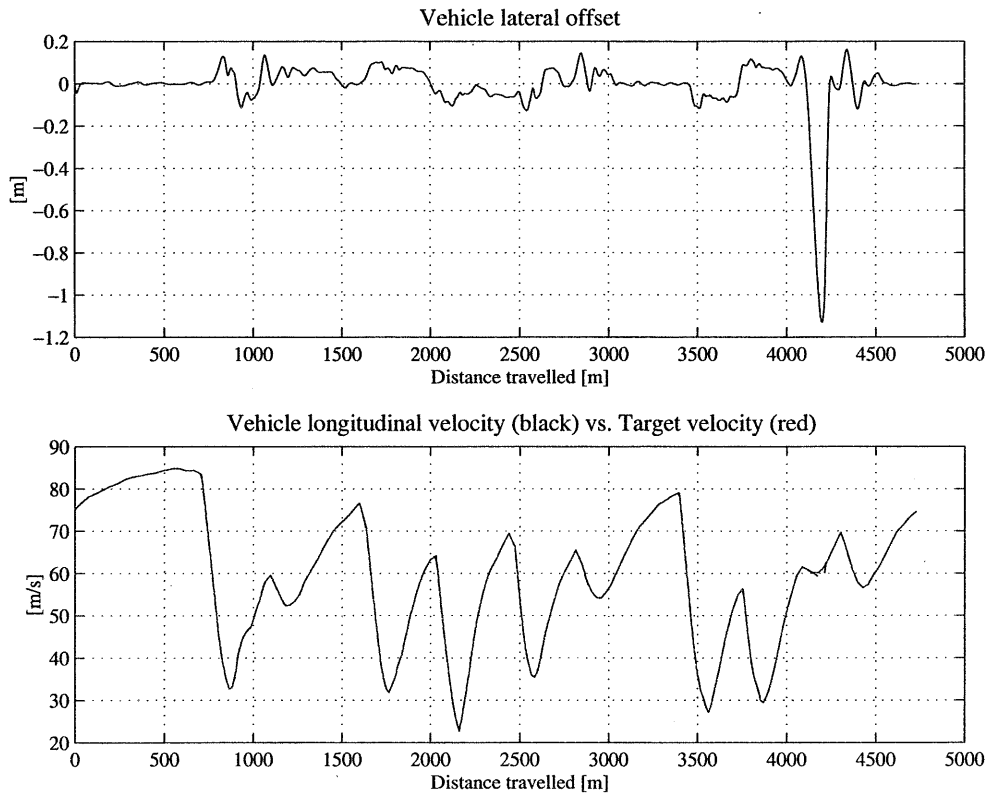


Figure 5.30 Path tracking error and vehicle longitudinal velocity.

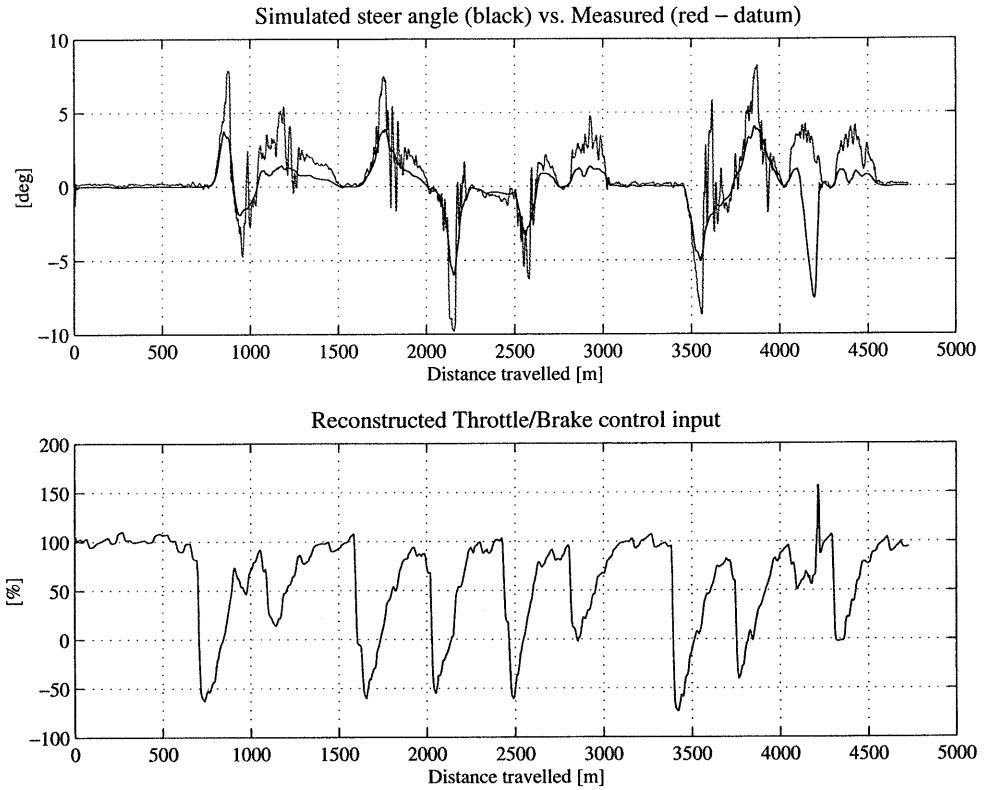


Figure 5.31 Vehicle lateral and longitudinal controls.

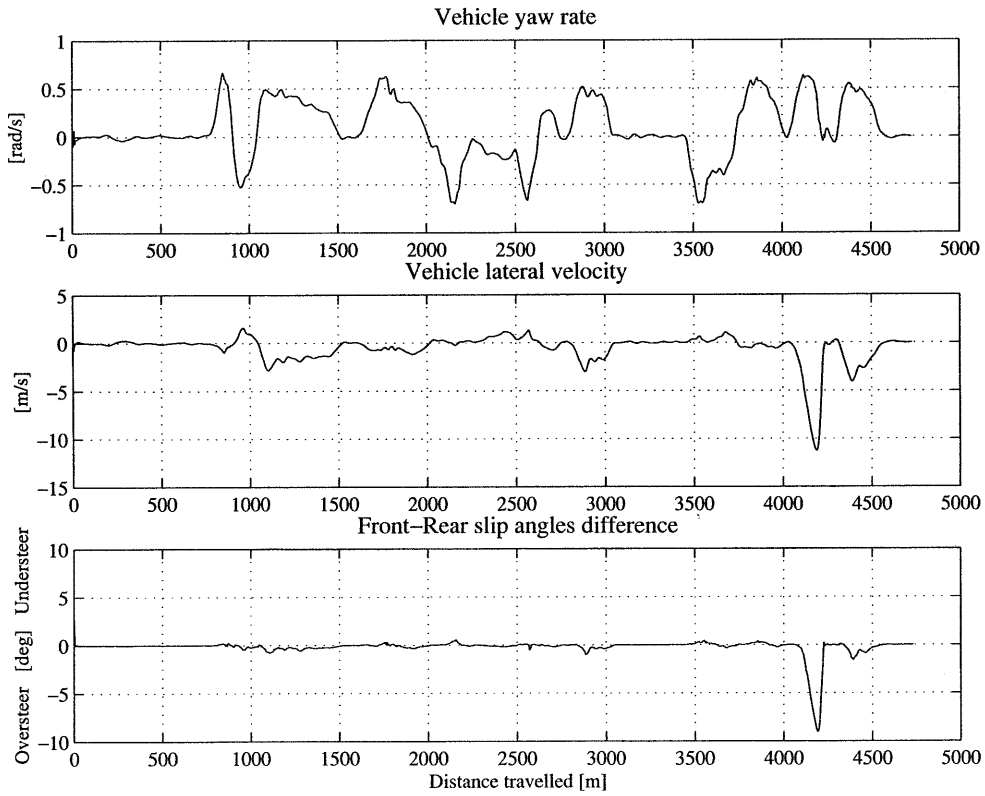


Figure 5.32 Vehicle yaw rate, lateral velocity and attitude.

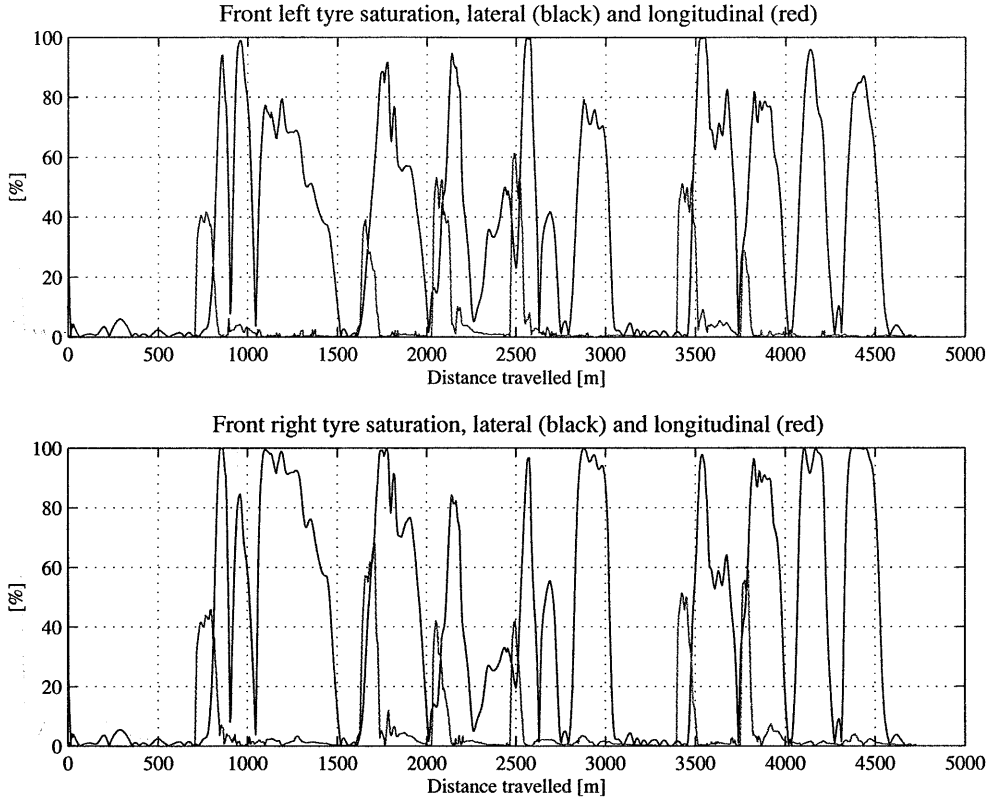


Figure 5.33 Front tyres lateral and longitudinal utilisation indexes.

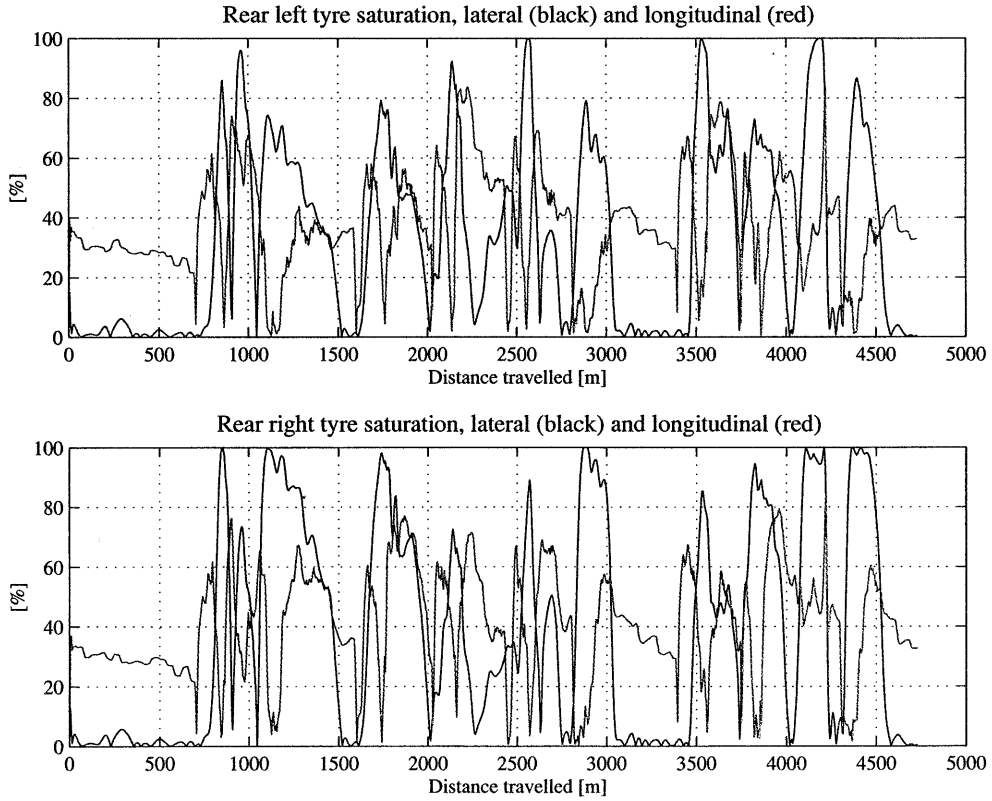


Figure 5.34 Rear tyres lateral and longitudinal utilisation indexes.

5.5. CONCLUSIONS

In this chapter the driver model described in the previous section has been applied to a non-linear, transient vehicle model representing a contemporary Formula One race car. A heuristic method for setting the steering control parameters has been outlined. Also the longitudinal speed control scheme has been described in detail.

By association of the steering controller with moderate manoeuvring and race car level operations of varied nature, the accuracy of the steering controller and its robustness to changes in vehicle or task have been demonstrated. Particularly important is the fact that no parameter changes to the driver model were required to complete any of the path following tasks.

An interesting correlation with the world of real driving emerged from the lap simulations with the understeer and the oversteer vehicles. While a tendency to understeer in road cars is seen as useful since it is much easier to control for the average driver, professional drivers usually prefer a vehicle whose behaviour is more biased towards oversteer. If a vehicle reaches the tyre lateral saturation firstly with the front axle, the total loss of directional control implies that the car can only depart from the intended trajectory and follow a path with larger cornering radius until the control is eventually regained, as long as there is sufficient road space available. If a vehicle, instead, reaches firstly the lateral limit with the rear tyres, the front tyres maintain directional control and a skilled driver can keep the car on the intended path. The same happens in the case of the path following simulation. With the understeer vehicle, when the front tyres reach the limit the car departs from the intended path and the driver model can do nothing until the conditions change in such a way that control is regained. Instead, with the oversteer vehicle, the driver model operates as a skilled driver and manages to maintain the intended course even when the rear tyres reach their lateral limit.

Chapter 6.

Lap Time Optimisation

The minimum race car lap time problem is formally defined here as a minimum time Optimal Control problem. Firstly, the mathematical model of the system, the performance measure and the constraints are specified according to the definitions which were given in chapter 2. Then, the choice of the solution method is discussed and the solution procedure is formally outlined, while the details of the implementation will be discussed in the next sections. The task here is to lay the basis for the development of a practical tool for the simulation and the optimisation of the performance of a circuit race car.

From this point of view, computational efficiency and robustness are seen as very important. If a lap time optimisation run takes a few hours of computing (considering the commonly available state-of-the-art hardware), the procedure will only be usable for "off line" (far from the race track) analysis. Although this would be of great value, for example, in the design phase of the racing car, if the computational time was limited well within an hour, possibly a few minutes, the simulation tool would be of aid also to race engineers during race weekends, when the time is very restricted. Furthermore the optimisation method should be compatible with the current techniques commonly in use for vehicle modelling. This often involves the use of multidimensional look-up tables with piece-wise continuous linear interpolation schemes, complex non-linear equations to model the tyre forces, e.g. the Magic Formula tyre model (Pacejka and Besselink, 1997), discontinuous step functions to represent for example the gear ratios, etc. Hence, the solution methods which allow to use the mathematical model of the vehicle "as it is", without the need of any symbolic manipulation, are advantageous with respect to the modelling issues. Finally, the accuracy of the results is important and the minimum tolerance required for the lap time simulation will determine a lower boundary for the computational time. At the highest level of motorsport, Formula One cars are developed in search of the ultimate performance and a few milliseconds make the difference between the first and the second row of the starting grid of a Grand Prix. Hence, when the task is to compare different vehicle configurations effectively, the ideal lap time simulation tool must consistently achieve comparable precision.

The problem of driving a car on its performance limits relates to the non-linear saturating properties of the tyre forces. In one of his books, Stirling Moss effectively describes driving a car as fast as possible through a corner as "*throwing a dart*". When a car approaches the limit of its cornering performance, the tyre forces available are mostly used to create the acceleration of the vehicle mass centre. Eventually, when the tyre forces saturate no margin is left for the driver to control the path of the vehicle without losing speed and consequently time. In many cases, though, when a driver makes a mistake and super-saturates the tyres, the consequences are much more

dramatic than losing time, because the space available to operate the vehicle, i.e. the road width, and apply the necessary corrections is very small in view of the speeds which are normally present.

These facts related to racing driving, i.e. the saturation of the tyre forces, the controllability of the vehicle on the limit of its performance and the narrow and awkwardly shaped road boundary constraints, translate into corresponding numerical problems for the solution of the theoretical optimal lap and will be key issues to consider when devising the solution strategy in order to guarantee the computational efficiency and the robustness of the lap time minimisation tool.

6.1. THE MINIMUM TIME MANOEUVRING OPTIMAL CONTROL PROBLEM

The setting up of the minimum time vehicle manoeuvring problem as one of Optimal Control Theory involves the definition of the vehicle equations of motion, the performance measure representing the manoeuvre time, the road boundary constraints and the control bounds.

6.1.1. The vehicle model equations

A general vehicle model is described by a set of first-order, non-linear differential equations, representing the rate of change of its p state variables $\mathbf{x}$ with respect to the time t:

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad \mathbf{x}(0) = \mathbf{x}_0 \qquad t \in [0, T] \qquad \text{Eq. 6.1}$$

Here, $\mathbf{u}(t)$ is a two-element vector function returning the vehicle lateral (i.e. the steer angle) and longitudinal (i.e. the throttle/brake control input) controls:

$$\mathbf{u}(t) = \{u_{st}(t) \quad u_{tb}(t)\} \qquad \text{Eq. 6.2}$$

Since in the minimum time Optimal Control problem T is not a constant, it is convenient to assume the vehicle travelled distance s, measured on the road centre line, as independent variable for the simulation instead of the time. For this purpose we shall use the time to distance scaling factor which was derived in § 4.2:

$$S_{CF} = \frac{(1 - d/r_t)}{V_{vx} \cdot \cos(\psi_v - \psi_t) - V_{vy} \cdot \sin(\psi_v - \psi_t)} \qquad \text{Eq. 6.3}$$

In the present case the reference line with respect to which the scaling factor is evaluated is the road centre line. This is described by its co-ordinates x_t and y_t with respect to a reference axis system fixed in space, the tangent angle ψ_t and the turn radius r_t (or the curvature k_t) as functions of the path length co-ordinate s, see figure 6.1. Hence, d in Eq. 6.3 represent the distance of the vehicle centre of gravity from the road centre line, whose expression reads:

$$d = (y_v - y_t) \cdot \cos(\psi_t) - (x_v - x_t) \cdot \sin(\psi_t) \qquad \text{Eq. 6.4}$$

Using Eq. 6.3, the vehicle model equations become:

$$\frac{d\mathbf{x}}{ds} = S_{CF} \cdot \mathbf{a}(\mathbf{x}(s), \mathbf{u}(s), s) = \bar{\mathbf{a}}(\mathbf{x}(s), \mathbf{u}(s), s) \quad \mathbf{x}(0) = \mathbf{x}_0 \quad s \in [0, S] \qquad \text{Eq. 6.5}$$

where S is the length of the road section that the vehicle has to traverse, which is therefore fixed.

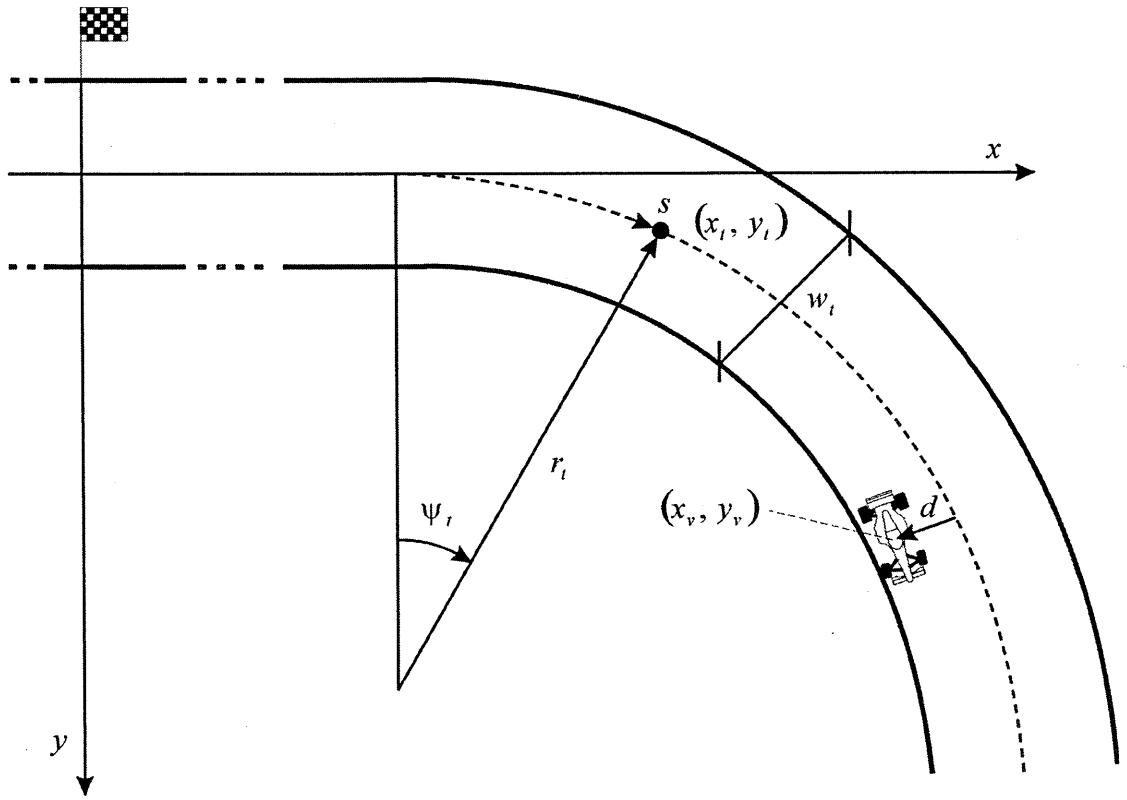


Figure 6.1 Circuit map and vehicle position from road centre line.

The accurate transformation of the problem from time to travelled distance domain achieved with the scaling factor, with respect to the fact that it takes into account the vehicle position and heading direction relative to the road centre line, is very important here. Two factors come into play when one is searching for the ideal vehicle trajectory which yields the minimum lap time. One, simply, is to maximise the vehicle velocity through the whole lap, which is though limited by the maximum engine power and the maximum tyre forces available, while the other is to minimise the distance travelled. The scaling factor embeds the latter information. Consider for example the vehicle depicted in figure 6.1. For a right hand corner, the curvature is positive. When the vehicle takes, as is expected, an inner line, the distance d from the road centre line is also positive and the numerator in Eq. 6.3 is smaller than one. Conversely, if the vehicle takes an outer trajectory, d becomes negative and the numerator in Eq. 6.3 is greater than one. Therefore, all other variables being equal, the scaling factor is consequently greater than in the previous case. Recalling that the scaling factor is defined as the ratio between the increment in the manoeuvre time dt

and the corresponding increment in travelled distance *ds*, measured along the road centre line:

$$S_{CF} = \frac{dt}{ds} \qquad \text{Eq. 6.6}$$

we may conclude that for the case of the vehicle travelling along an outer line the greater scaling factor indicates that the vehicle takes more time to traverse the same road distance compared with the case of the vehicle taking correctly the inner line, because the actual travelled distance is greater. Hence the accuracy of the scaling factor is fundamental in order to obtain realistic racing lines.

6.1.2. The performance measure

The performance measure is the time that the vehicle takes to traverse a specified road section or an entire lap of a circuit. The time to distance scaling factor offers a straightforward way to evaluate the manoeuvre time. From now on, let us adjoin one more state variable x_{p+1} which satisfies:

$$\dot{x}_{p+1}(s) = S_{CF}(s) \qquad x_{p+1}(0) = 0 \qquad s \in [0, S] \qquad \text{Eq. 6.7}$$

According to the definition of the scaling factor given in Eq. 6.6, the added state variable represents the time elapsed from the beginning of the manoeuvre. Thus, the performance measure simply reads:

$$J = x_{p+1}(S) \qquad \text{Eq. 6.8}$$

6.1.3. The road boundary constraints

The road boundary constraints determine a very narrow and awkwardly shaped feasible region for the vehicle trajectory. The mathematical description of these constraints is then crucial and may affect the computational efficiency of the optimisation procedure. A few options are possible and figure 6.2 shows the two main approaches.

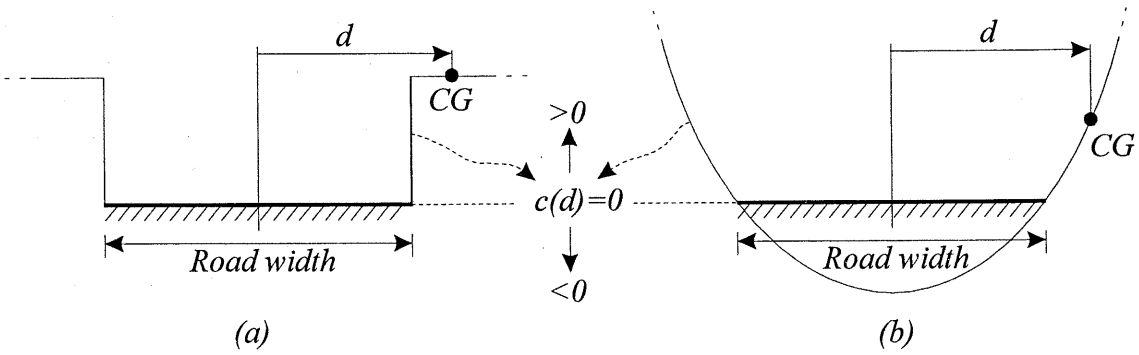


Figure 6.2 Road boundary constraint definition.

Consider firstly figure 6.2 (a). A function $c(d)$ of the distance d of the vehicle centre of mass *CG* from the road centre line is defined in such a way that it returns 0 when the

vehicle position lies on the road surface, while it returns values greater than 0 when the vehicle position lies off the road:

$$\begin{aligned} c(d) &= 0 \quad when \quad |d| \le \frac{1}{2} \cdot w_t \\ c(d) &> 0 \quad when \quad |d| > \frac{1}{2} \cdot w_t \end{aligned} \qquad \text{Eq. 6.9}$$

The function $c(d)$ needs not necessarily to be discontinuous as is shown in figure 6.2 (a). Rather, a linear increment of the constraint value may be obtained by defining the second of Eq. 6.9 as follows:

$$c(d) = |d| - \frac{1}{2} \cdot w_t \quad when \quad |d| > \frac{1}{2} \cdot w_t \qquad \text{Eq. 6.10}$$

By integrating the function $c(d)$ over the travelled distance the measure of the overall road boundary constraint violation may be obtained:

$$c = \int_0^S c(d) ds \qquad \text{Eq. 6.11}$$

The advantage of this approach is that one single constraint is sufficient. In fact, imposing that c be equal to 0 enforces the vehicle trajectory to lie within the road boundaries for the whole manoeuvre.

This method was employed successfully by (Hendrikx et al., 1996) to find the optimal vehicle trajectory for simple manoeuvres, e.g. a single lane change. However, when considering a more complex road geometry, this approach is not ideal. The nature of the minimum time manoeuvring problem implies that the vehicle uses all the road width available whenever this allows to achieve greater speeds. Therefore, many sections of the optimal vehicle trajectory will lie right on the road boundaries, where the transition of the function $c(d)$ from zero to non-zero values occurs. According to the definitions given by the first of Eq. 6.9 and Eq. 6.10, the function $c(d)$ will be continuous across the road boundary, but not differentiable. Then the constraint violation measure c obtained with Eq. 6.11, because of the integration procedure, will be continuous and differentiable once. Its curvature, though, will not be continuous across the road boundaries, where the vehicle trajectory is often expected to lie at the solution. This constitutes a problem for all the optimisation methods which use derivatives of the objective and constraint functions to achieve their goal.

A more efficient definition for the road boundary constraints is that shown in figure 6.2 (b). A continuous and twice-differentiable function $c(d)$ of the distance d of the vehicle centre of mass from the road centre line is defined as follows:

$$c(d) = \left(\frac{2 \cdot d}{w_t} \right)^2 - 1 \qquad \text{Eq. 6.12}$$

Eq. 6.12 returns positive values when the vehicle position lies off the road boundaries, zero when it lies exactly on one of the road boundary, and a minimum value equal to –1 when the vehicle position coincides with the road centre line. Hence, the condition for the vehicle to stay on the road is that $c(d)$ is non-positive. In this case, though, it is not possible to define a unique measure for the overall road boundary constraint violation by integrating $c(d)$ over the length of the manoeuvre. Instead, Eq. 6.12 must be evaluated for a sufficient number of *check points* distributed along the road section in such a way that if the vehicle trajectory satisfies the road boundary constraints at the check points, then the road boundaries will not be violated anywhere else. In order to define these check points, we shall exploit once more the nature of the minimum time manoeuvring problem.

Consider figure 6.3, which shows qualitatively the racing line through the first two corners of the Suzuka circuit. In order to identify how many road boundary check points are needed and their locations, we may observe that while the minimum time manoeuvring task implies that the vehicle uses all the road width available in certain road sections to achieve maximum speeds, in other sections the vehicle takes the shortest possible trajectory in such a way to minimise the actual travelled distance. Figure 6.3 shows that the initial segment of the vehicle trajectory needs not to be constrained, as the vehicle proceeds along a straight line until shortly before turn 1, where it uses all the space available on its left hand side in order to maximise the speed when going into the corner (c_1). Then, the vehicle trajectory is tangent to the right hand side road boundary at the apex of turn 1 (c_2), and successively widens towards the left hand side of the track before turn 2 (c_3). Finally, the apex of turn 2 is the next active boundary (c_4), followed by the left hand side of the track at the exit of turn 2 (c_5).

Even though we are able to identify sections of the road where the boundaries effectively constrain the vehicle trajectory, we are not able to specify precisely where the check points should be located. The strategy adopted consists then in monitoring the distance of the vehicle centre of mass from the road centre line within specified sections of the road, and take its maximum value to evaluate Eq. 6.12.

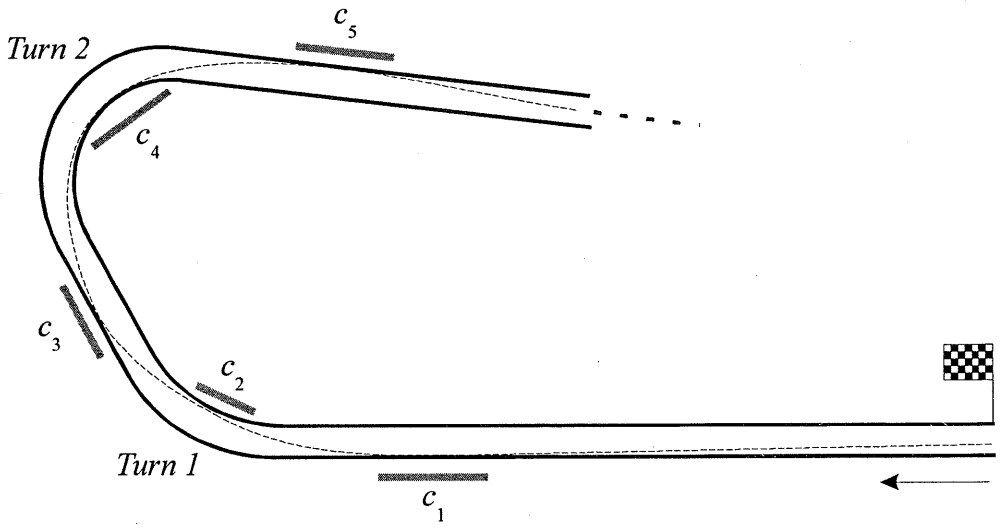


Figure 6.3 Vehicle racing line and binding road boundary constraints.

Referring to figure 6.4, which shows a detail of the racing line depicted in figure 6.3, we may write for the generic i^{th} road boundary constraint c_i:

$$d_{MAX} = \max(d(s)) \qquad s \in [s_{i\_ini}, s_{i\_end}] \qquad \text{Eq. 6.13}$$

and:

$$c_i = \left(\frac{2 \cdot d_{MAX}}{w_t} \right)^2 - 1 \qquad \text{Eq. 6.14}$$

The road sections where the vehicle position is to be monitored and the constraints evaluated are usually defined within the model of the circuit by specifying the initial and final path co-ordinates s_{i_ini} and s_{i_end}, as we shall see later on. Although this strategy involves some judgement when modelling the circuit, it allows to accurately impose the road constraints by using smooth functions and this is advantageous compared with the other method described earlier. Finally, the road boundary constraints may be multiplied by scaling coefficients in order to give them greater weight in the optimisation process in such a way that one may distinguish the case of the unforgiving walls of Monaco from the more forgiving kerbs of Silverstone.

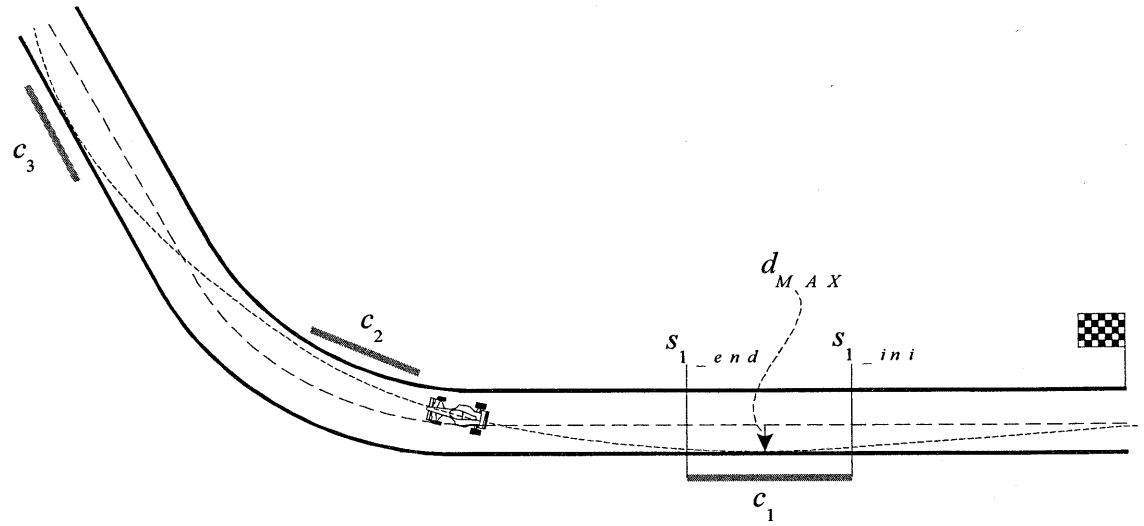


Figure 6.4 Identification of road boundary check points.

6.1.4. The control bounds

Constant control bounds are applied in order to restrict the control variables within values which are physically meaningful for the vehicle model. For the vehicle lateral control u_{st}, the steer angle is limited to a maximum value which is set on the real car by steering stops in the mechanism. For the vehicle longitudinal control, the variable u_{tb} represents both the throttle and the brake control input. The longitudinal control variable

is allowed to vary within the interval [+1, -1]. When positive, the control variable represents the throttle aperture, with $u_{tb} = 1$ meaning the full throttle aperture, and its value is used in the mathematical model of the powertrain to evaluate the driving torque. Instead, when negative, the control variable represents the brake input, and its value simply factors the maximum braking torque available. Hence, $u_{tb} = -1$ corresponds to the maximum braking torque applicable to the wheels.

6.1.5. Problem statement

The task in the minimum time manoeuvring Optimal Control problem is to find an admissible vehicle lateral and longitudinal control history $\mathbf{u}(s)$, i.e. a control history which does not exceed the specified control bounds, which causes the vehicle to follow a feasible trajectory $\mathbf{x}(s)$, i.e. a trajectory which does not violate the road boundary constraints, in as short a time as possible:

$$
\begin{aligned}
&\min_{\mathbf{u}} \quad J \\
&\textit{subject to}: \\
&\textit{Vehicle equations}: \quad \frac{d\mathbf{x}}{ds} = \bar{\mathbf{a}}(\mathbf{x}(s), \mathbf{u}(s), s) \qquad \mathbf{x}(0) = \mathbf{x}_0 \\
&\textit{Road constraints}: \quad c_i \le 0 \qquad \forall i \\
&\textit{Control bounds}: \quad \mathbf{u}_L \le \mathbf{u}(s) \le \mathbf{u}_U \\
&\textit{for all} \quad s \in [0, S]
\end{aligned}
\qquad \text{Eq. 6.15}
$$

6.2. THE SOLUTION METHOD: DIRECT OR INDIRECT APPROACH?

The methods for solving Optimal Control problems were reviewed in chapter 2. Two distinct classes of methods were presented, i.e. direct and indirect methods. Indirect methods rely on the application of the theory of Calculus of Variation, i.e. the Pontryagin's Minimum Principle. Direct methods, instead, aim to solve an Optimal Control problem by converting the original continuous problem into a discretised problem and applying mathematical programming techniques. In this work we aim to solve the minimum lap time optimisation problem by applying a direct method.

Optimisation is still often seen as a field for specialists, as there are not many general software tools available and, more important, they are difficult to integrate with the common design and simulation tools in use by engineers. Our intent here is to develop the lap time optimisation program in order to make it more user oriented. Ideally, the vehicle model may be given a modular structure in such a way that the end user may easily extend the function library by adding different models for the various components, e.g. power train model, tyre model, aerodynamic forces model etc., with the maximum freedom for modelling the systems and without the need for further manipulation of the model equations. Direct solution methods offer the ideal opportunity to achieve this goal.

Another aspect in favour of direct methods is the greater freedom in defining and including state and control constraints in the optimisation problem compared with indirect methods. As was outlined in §2.3, the solution of constrained optimisation problems by indirect methods either requires the use of specially designed algorithms or the use of penalty functions to include the constraints in the objective function. For the case of the minimum lap time optimisation problem the definition of the road boundary constraints is crucially important, as was described above. A strategy to deal with them has been devised which is thought to be ideal both from the point of view of the numerical efficiency as well as the accuracy. Such strategy may be applied straightforwardly with the direct solution methods.

The lateral and longitudinal controls of a racing car feature very rapid variations among nearly constant or slowly varying parts. Particularly with regard to the longitudinal control, when approaching a turn after a long and straight road section the switch from maximum driving into braking is ideally discontinuous. Indirect solution methods, which deal with the continuous optimal control problem, are not directly applicable in this case. The strategy would involve dividing the vehicle trajectory in piecewise continuous arcs whose junctions have to be located where the control switch occurs. The location of the junctions, though, is not known in advance and this adds further complications. Conversely, with direct methods the discretisation of the control history allows to model any type of control law straightforwardly.

Direct methods have then many practical advantages which are very attractive when optimisation techniques have to be applied to engineering problems. For this reason the research in the field of computational optimal control in recent years favours direct methods. Well established numerical libraries are now available for solving Non-Linear Programming problems[1] (Kraft 1994, Gill et al., 1997) and these methods have been successfully applied to a variety of engineering tasks, including for example aerospace and robotic trajectory planning.

The disadvantage with direct methods is that because of their discrete nature they only return approximate solutions. The user must be aware of the influence of the discretisation scheme on the solution. The appropriate strategy is to progressively refine the discretisation of the problem until sufficient accuracy in the solution is achieved. It is important to remember, though, as was pointed out in chapter 2, that even if one uses an indirect method some sort of discretisation will be required in order to solve the resulting non-linear two-point boundary problem on a digital computer. Hence, indirect methods are not completely immune from similar problems.

6.3. THE DISCRETISATION SCHEME

The transcription of an Optimal Control problem into a Non-Linear Programming problem involves firstly the discretisation of the control history. In order to do so, we shall assume that the control action can only be adjusted by the optimiser at a number of fixed positions along the road, while the intermediate values are determined by means of interpolation techniques. The instances where the control parameters are located form

[1] The web site: `www-fp.mcs.anl.gov/otc/Guide/SoftwareGuide/index.html` is a convenient link to optimisation software available.

the *communication grid*, which may be defined as the interface between the optimiser and the continuous control history. Figure 6.5 shows qualitatively the communication grid. Each control parameter is identified by its path co-ordinate and the actual control value, i.e. for the i^{th} control point:

$(s_{sti}, u_{sti}) \rightarrow$ steer angle control point
$(s_{tbi}, u_{tbi}) \rightarrow$ throttle / brake control point

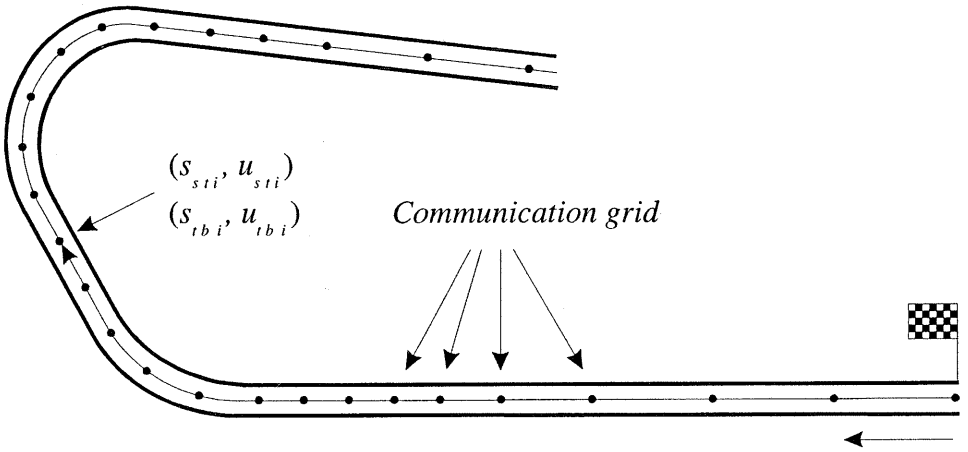


Figure 6.5 Control parameterisation: the communication grid.

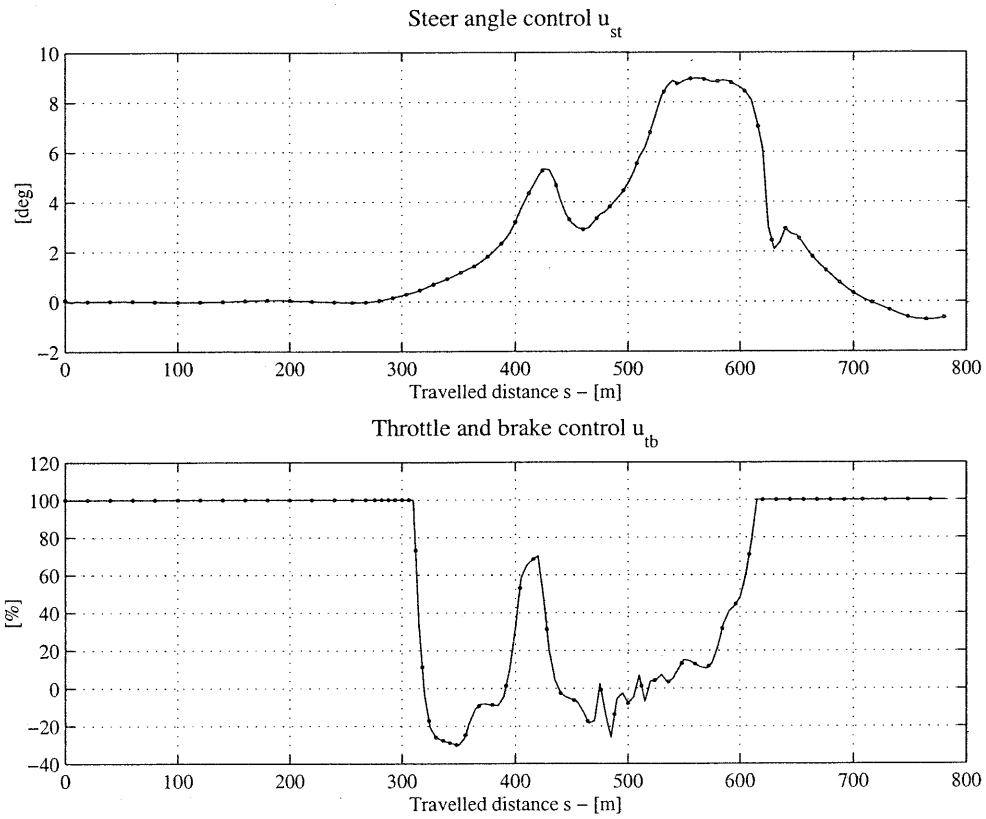


Figure 6.6 Variable density communication grids for vehicle controls.

Separate communication grids may conveniently be specified for the lateral and the longitudinal vehicle controls. Furthermore, it is advantageous to increase the density of the communication grid where the rate of change of the control is expected to be greater, e.g. at braking points, while fewer points may be sufficient where the controls are nearly constant, hence reducing the problem size. Figure 6.6 illustrates this idea qualitatively by showing the discretisation scheme of the vehicle controls for the two turns of the Suzuka circuit section depicted in figure 6.5.

The next necessary step in the transcription of the minimum lap time Optimal Control problem into a Non-Linear Programming problem is the discretisation of the vehicle state trajectory into independent segments. If the vehicle trajectory was treated as being continuous, the maximum length of the manoeuvre which could be efficiently optimised would be limited by the coupling which arises between the early controls and the later segments of the trajectory. Two main problems would consequently affect the optimisation process, and these are described in figures 6.7 and 6.8.

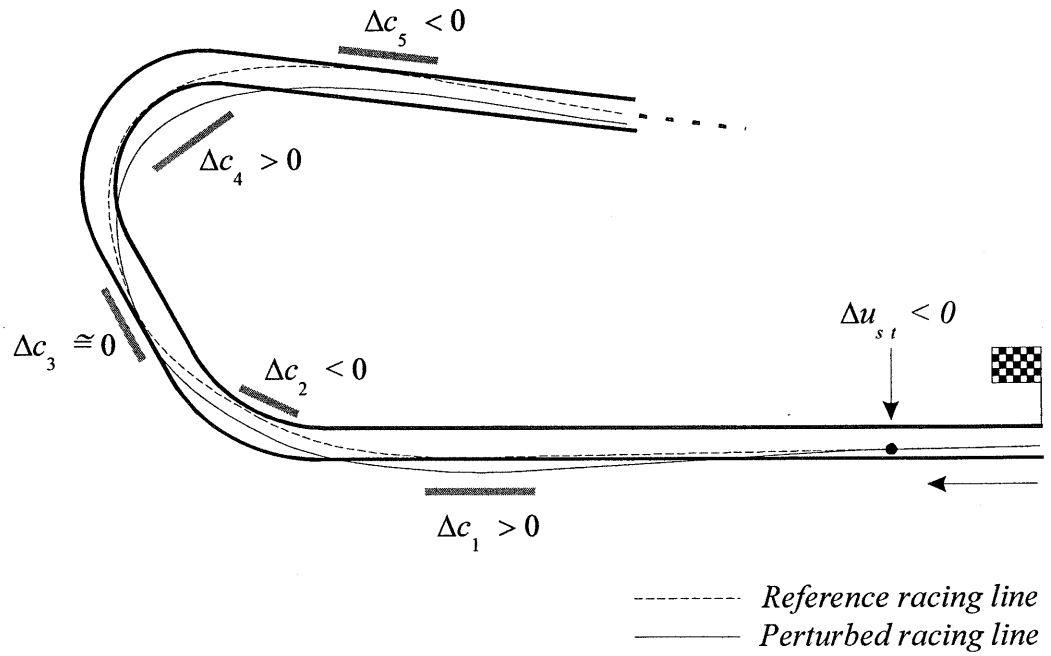


Figure 6.7 Road boundary constraint variations resulting from early control perturbations.

Consider figure 6.7 first. The dashed line represents the racing line at some intermediate stage of the optimisation. At any iteration the optimisation algorithm needs to evaluate the sensitivity of the objective and the constraints with respect to all the control parameters in order to improve the control array. Let us imagine that in order to find such sensitivities a small negative perturbation has been given to an early steer angle control point (according to the sign convention this corresponds to steering towards the left hand side of the road). From the varied control point onwards, the continuous line represents qualitatively the variation of the racing line as a result of such single perturbation. Clearly, such variation influences differently the five road boundary check points which are marked in figure 6.7. Particularly, the first and fourth road constraints increase, the second and the fifth decrease while the third remains nearly unchanged. The optimiser would then obtain conflicting sensitivity information

which ultimately would not allow to adjust the early control parameters effectively. Furthermore the constraints which are further away would be affected to a larger extent by the early control perturbation. As a consequence, we would also obtain badly scaled derivative information. For a real driver this would correspond to the unrealistic situation where he/she would decide the vehicle control when crossing the start/finish line by giving the highest priority to controlling the vehicle in the last corner of the circuit!

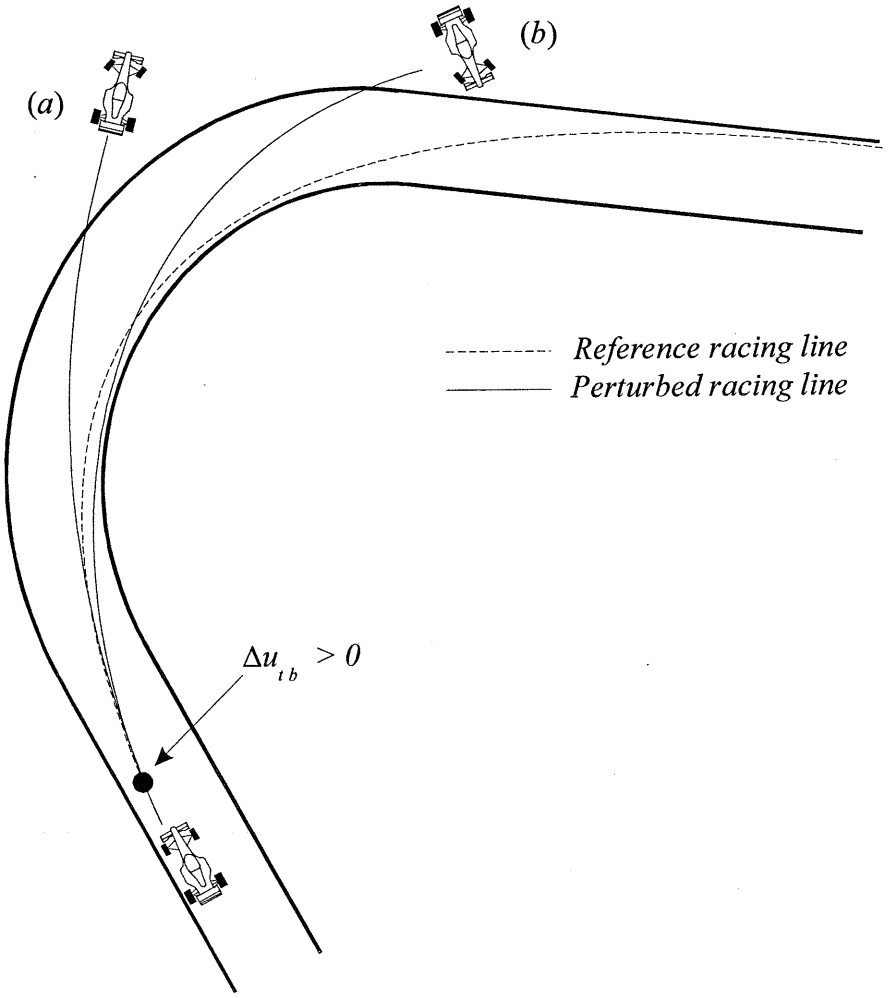


Figure 6.8 Vehicle controllability issues as a consequence of control variation when near the performance limit.

The second fundamental aspect to consider for the state trajectory discretisation relates to the saturation of the tyre forces, see figure 6.8. Finding the driving strategy which yields minimum manoeuvre time involves using all the tyre forces available. Inevitably, the optimiser has to deal with the problem of controlling the car on the limit as well as a real driver has. Figure 6.8 illustrates the problem. Here, the dashed line is a sub-optimal racing line, i.e. a racing line which relates to a manoeuvre very near to the limit of tyre adhesion. A small perturbation is then applied to an early longitudinal control point by increasing the thrust and consequently the velocity. As the vehicle was already very close to the limit, the manoeuvre which results after applying the control

variation may exceed its performance boundary and result in one of the two situations depicted in figure 6.8, depending on whether the front tyres (a) or the rear tyres (b) reach the limit first. The second case, which corresponds to the vehicle becoming divergently unstable, is the more difficult to deal with. Obviously, if the vehicle spins off the road we are not able to obtain meaningful values for the objective and constraints. However, we do not need to get that far in order to experience problems during the optimisation process. In fact, when the vehicle is very close to the limit, even though the simulation may still be accomplished, the sensitivity of the objective and the constraints, particularly with respect to early controls, may reveal that the vehicle is very close to becoming unstable. In other words, when the vehicle reaches the limit of adhesion of the tyres, the sensitivity of the trajectory with respect to the control parameters may increase significantly, hence yielding very badly scaled problem derivatives. The result is very slow convergence when near the optimum or even failure to converge.

The key issue in order to avoid both problems which have been described here is then to divide the trajectory into completely de-coupled segments, in such a way that each control parameter can only influence a short part of the trajectory ahead. Particularly with regard to the vehicle stability problem, the length of the trajectory segments must be kept sufficiently short so that, according to the vehicle dynamic properties, *the spin cannot occur within the distance of the segment.*

Among the direct transcription methods which were described in chapter 2, the parallel shooting method (Enright and Conway, 1990, Betts and Huffman, 1991) is ideally suited to the discretisation scheme which is proposed here. Also the formulation of the road boundary constraints which was devised earlier in this chapter may nicely be integrated with this discretisation scheme. Ideally, we may think of road segments with "active" road boundary check points as opposed to road segments where the road boundaries need not to be enforced.

As a consequence of the total de-coupling of the trajectory segments, each control parameter will only affect a small number of the problem constraints. This results in a significant degree of sparsity in the Jacobian which may be advantageously exploited by the optimisation algorithm. Finally, this discretisation scheme opens the possibility for the parallel computing of the objective, the constraints and their derivatives, as a consequence of the independence of the trajectory segments.

6.4. THE PARALLEL SHOOTING ALGORITHM

The parallel shooting algorithm was outlined in §2.4. In this section we shall extend the definition for the case of the minimum lap time optimisation problem.

Consider firstly the control parameterisation. For generality, we shall define two different communication grids for the vehicle lateral and longitudinal controls. For the steer angle control, we shall consider n_{st} control intervals as follows:

$$\mathbf{s}_{\mathbf{stn}} = \{0 = s_{st0} < s_{st1} < s_{st2} < \cdots < s_{stn-1} < s_{stn} = S\} \qquad \text{Eq. 6.16}$$

The corresponding vector of steer angle control parameters then reads:

$$\mathbf{u}_{\mathbf{stn}} = \{u_{st0}\ u_{st1}\ u_{st2}\ \cdots\ u_{stn-1}\ u_{stn}\} \qquad \text{Eq. 6.17}$$

Similarly, for the vehicle longitudinal control we shall define n_{tb} control intervals:

$$\mathbf{s}_{\mathbf{tbn}} = \{0 = s_{tb0} < s_{tb1} < s_{tb2} < \cdots < s_{tbn-1} < s_{tbn} = S\} \qquad \text{Eq. 6.18}$$

and the corresponding longitudinal control parameters:

$$\mathbf{u}_{\mathbf{tbn}} = \{u_{tb0}\ u_{tb1}\ u_{tb2}\ \cdots\ u_{tbn-1}\ u_{tbn}\} \qquad \text{Eq. 6.19}$$

For simplicity of exposition, we shall combine lateral and longitudinal control parameters into one single control array, whose dimension will be $n = n_{st} + n_{tb}$:

$$\mathbf{u}_{\mathbf{n}} = \mathbf{u}_{\mathbf{stn}} \cup \mathbf{u}_{\mathbf{tbn}} \qquad \text{Eq. 6.20}$$

Analogously, we shall combine the two communication grids defined in Eqs. 6.16 and 6.18 into one single array:

$$\mathbf{s}_{\mathbf{n}} = \mathbf{s}_{\mathbf{stn}} \cup \mathbf{s}_{\mathbf{tbn}} \qquad \text{Eq. 6.21}$$

It is assumed that the control input to the vehicle may only be varied by the optimiser at the nodes of the communication grids, while the intermediate values are estimated by means of interpolation techniques. Hence the control parameters uniquely define the control history. Formally, we may write:

$$\mathbf{u}(s) = \text{interp}(\mathbf{s}_{\mathbf{n}}, \mathbf{u}_{\mathbf{n}}, s) \quad \Rightarrow \quad \mathbf{u} = \mathbf{u}(\mathbf{u}_{\mathbf{n}}) \qquad \text{Eq. 6.22}$$

Next, consider to divide the state trajectory in m segments by defining another mesh of points as follows (see figure 6.9):

$$\mathbf{s}_{\mathbf{m}} = \{0 = s_0 < s_1 < s_2 < \cdots < s_{m-1} < s_m = S\} \qquad \text{Eq. 6.23}$$

To indicate the values of the state variables at the i^{th} node, we shall use the following notation:

$$\mathbf{x}_i = \mathbf{x}(s_i) \qquad i = 0,\ldots,m \qquad \text{Eq. 6.24}$$

The values of the state variables at the beginning of each segment constitute the vector of the independent state parameters which the optimisation algorithm is allowed to adjust in order to restore the continuity of the state trajectory across the junctions:

$$\mathbf{x}_{\mathbf{m}} = \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m-1}\} \qquad \text{Eq. 6.25}$$

If there are p state variables, the vector $\mathbf{x_m}$ will have $p \times m$ elements. Finally, we may combine the control parameters with the state parameters to define the vector $\mathbf{y}$ of all the independent optimisation variables, whose dimension will be $n + p \times m$:

$$\mathbf{y} = \mathbf{u}_n \cup \mathbf{x}_m \qquad \text{Eq. 6.26}$$

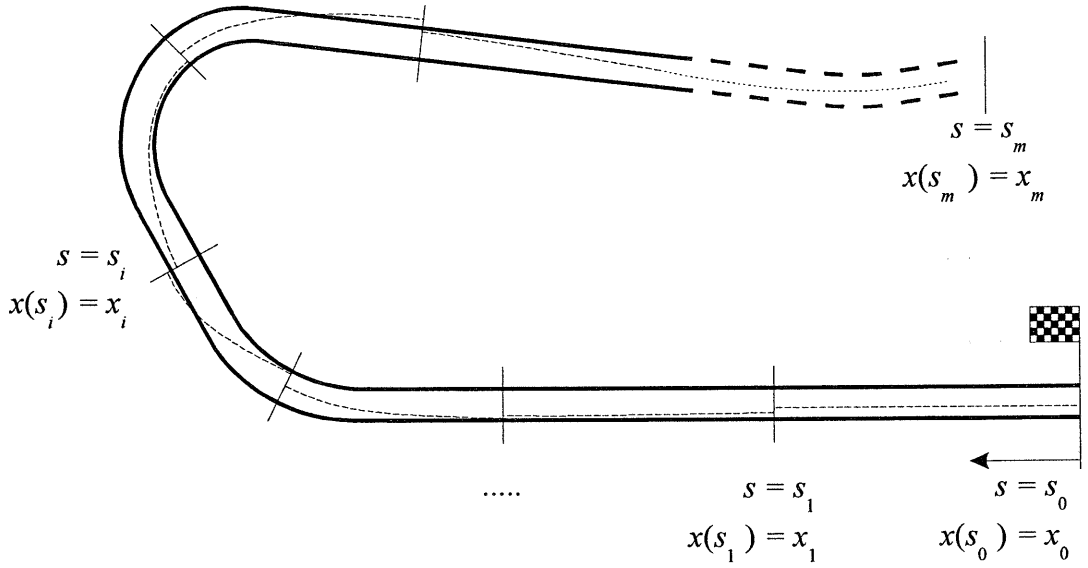


Figure 6.9 Trajectory discretisation scheme.

The trajectory within each segment is evaluated by solving the following initial value problem with a suitable method, e.g. Runge-Kutta:

$$\frac{d\mathbf{x}}{ds} = \bar{\mathbf{a}}(\mathbf{x}(s), \mathbf{u}(s), s)$$

$$with: \quad \mathbf{x}(s_{i-1}) = \mathbf{x}_{i-1} \qquad s \in [s_{i-1}, s_i] \qquad \text{Eq. 6.27}$$

$$for: \quad i = 1, \ldots, m$$

If the road boundary check point is active within the i^{th} segment, the distance of the vehicle centre of mass from the road centre line is monitored during the solution of the above initial value problem, and its maximum distance from the road centre line is taken to evaluate the constraint defined by Eq. 6.14. In general, not all the trajectory segments will need active road bounds; hence the number of check points will be $m_c < m$.

The fact that the integration of the vehicle model equations is restarted using the independent state parameters $\mathbf{x_m}$ realises the complete de-coupling of the segments. The continuity of the state trajectory across the junctions needs then to be re-established by imposing sufficient constraints. For each of the interior nodes of the state discretisation mesh, we shall define the vector of defects as follows:

$$\xi_i = \mathbf{x}(s_i) - \mathbf{x}_i \qquad i = 1, \ldots, m-1 \qquad \text{Eq. 6.28}$$

If a closed lap of a circuit is considered, then one extra defect must be defined in order to account for the continuity of the trajectory at the start-finish line:

$$\xi_m = \mathbf{x}(s_m) - \mathbf{x}_0 \qquad \text{Eq. 6.29}$$

The continuity is therefore established by imposing that all the defects be equal to zero at the solution. Finally, the total number of the problem constraints will be the sum of the number of the road boundary check points and the number of defects, i.e. $m_c + p \times m$.

The state trajectory within each segment is univocally determined by the independent optimisation variables $\mathbf{y}$. Therefore we may formally write:

$$\mathbf{x} = \mathbf{x}(\mathbf{y}) \qquad \text{Eq. 6.30}$$

In turn, the state trajectory allows to evaluate the objective function and the road boundary constraints according to Eqs. 6.8, 6.13 and 6.14. Hence, the objective and constraint functions may also be expressed directly as functions of the independent optimisation variables, and the original Optimal Control problem 6.15 may be converted into the following Non-Linear Programming problem:

$$\begin{aligned} &\min_{\mathbf{y}} \quad J(\mathbf{y}) \\ &\textit{subject to}: \\ &\textit{Continuity constraints}: \quad \xi_i(\mathbf{y}) = 0 \qquad i = 1,\ldots,m \\ &\textit{Road constraints}: \quad c_i(\mathbf{y}) \le 0 \qquad i = 1,\ldots,m_c \\ &\textit{Bounds}: \quad \mathbf{y}_L \le \mathbf{y} \le \mathbf{y}_U \end{aligned} \qquad \text{Eq. 6.31}$$

Suitable bounds apply to all the independent optimisation variables, including the independent state parameters $\mathbf{x_m}$, in order to ensure the feasibility of the starting point of each trajectory segment.

6.5. CONCLUSIONS

In this section the strategy for solving the minimum race car lap time optimisation problem has been devised. The analysis of the nature of the problem has led to a discretisation scheme which restricts the range of influence of each control point within short segments of the trajectory, hence avoiding the unnecessary and disadvantageous coupling between early control actions with later parts of the vehicle manoeuvre. The discretisation scheme matches perfectly with the parallel shooting method which was introduced in chapter 2, and it will be adopted for the future developments of the lap time optimisation program. The only alternative which would have ensured the required de-coupling with the added advantage of not requiring the solution of the differential equations of motion of the vehicle was the direct collocation method. However, such method would have required a very dense discretisation mesh for the state trajectory, since the typical integration step size for the vehicle model is of the order of 0.1 m or less, hence increasing enormously the problem size.

So far we have only discussed the principles which must be used as a guide to develop the discretisation scheme, but we have not discussed how many control points

are actually needed or how long should the trajectory segments be. In fact, the optimal discretisation scheme may only be found by successive trials, progressively refining the mesh until satisfactory behaviour and accuracy for the lap time optimisation program is achieved. However, we may anticipate at least some correlation between the preview distance L_p, which was introduced in chapter 4 regarding the driver model, and the length of the trajectory segments. Our heuristic interpretation of the preview distance is that for a given control input applied at a generic location s, the vehicle response will be accomplished within the interval $[s, s + L_p]$. It is then reasonable to assume that the length of the trajectory segments should be in the same region as the distance L_p, which, for the case of a circuit race car, was the distance travelled by the vehicle in one second.

The next phase of the development of the lap time optimisation program is then to use a Sequential Quadratic Programming (SQP) algorithm to solve problem 6.31 and set up increasingly more complex optimisation tasks in order to find the optimal discretisation of the problem which ensures robust behaviour for all anticipated vehicle configurations. However, the SQP algorithm requires the first order derivatives of the objective function and of the constraints with respect to all the independent optimisation variables. For non-linear, large-scale optimisation problems this is crucially important. Obtaining fast and accurate derivatives may be the key to success or the doorway to failure for the type of problem which is being studied here. Therefore the subject deserves the entire next chapter.

Chapter 7.

Evaluating Derivatives

"Dear Daniele! Hopefully this will cause a chain rule reaction!"

(Hand written by Prof. Andreas Griewank on my copy of his book on Automatic Differentiation, on 12/07/2000)

For large-scale, non-linear optimisation problems, such as the minimum lap time optimisation problem, evaluating derivatives is the most expensive task of the overall calculation. Furthermore, obtaining accurate values for the derivatives is crucial in order to ensure a robust behaviour for the optimisation algorithm and sufficient accuracy for the optimal solution.

Whenever we need to differentiate an algebraic function two methods are readily available: symbolic differentiation and finite difference approximation techniques (numerical differentiation). When applicable, symbolic differentiation is certainly the best approach in terms of computational efficiency and accuracy. However, as the complexity of the mathematical models which are used for engineering problems grows, symbolic differentiation becomes less practical and, ultimately, may not be feasible. On the other hand, finite difference techniques are always easily applicable, since the mathematical model is treated as a black box which simply connects dependent and independent variables. However, the evaluation of the gradient of a function of n independent variables by means of finite difference techniques requires at least $n + 1$ function evaluations. Since for optimisation problems the number of independent variables may easily be as large as several thousands, numerical differentiation becomes extremely time-consuming and may only be applied if the problem has a suitable sparse nature. Furthermore, numerical differentiation is often inaccurate because of truncation and round off errors.

A third method for the differentiation of functions defined by computer programs, known as *Automatic Differentiation* (or *Algorithmic Differentiation*) is available. Automatic Differentiation (AD) is a programming technique for obtaining derivatives of numerical functions to the same order of accuracy as the function values themselves, but without the labour of forming explicit symbolic expressions for the derivative functions. AD works by repeated use of the chain rule, but applied to numerical expressions rather than to symbolic values. There are two main approaches to implement AD. With the *forward accumulation technique* each program variable is associated to a vector containing the partial derivatives of that variable with respect to the independent variables. As the function evaluation proceeds, every time a program variable is involved in an algebraic operation its corresponding vector is updated and at the end it will contain its gradient. Conversely, the *reverse accumulation technique* builds a

computational graph for the function evaluation, with one node for each value ever held by a program variable, and associates with each node an *adjoint* value containing the partial derivative of the function value with respect to the node. The adjoint values are computed in reverse order to the function evaluation, hence the name "reverse accumulation". Reverse accumulation is of particular interest when the gradient of a scalar function is required. In this case the theory states that the gradient may be obtained at a computational cost of three function evaluations in total, regardless of the number of independent variables and of the function complexity (Griewank, 1989).

In the framework of the lap time optimisation program, symbolic differentiation is not regarded as suitable, as one of the aims of this work is to allow maximum freedom in modelling the vehicle as well as the circuit. Therefore, it will not be considered further. The next paragraph gives a short insight into the use of numerical differentiation techniques, while the rest of the chapter focuses on AD, starting with a brief overview of how it works, describing the forward and the reverse accumulation techniques by means of simple examples. The keen reader who is interested in the theoretical background of AD and in the details regarding its implementation may refer to the recently published book *Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation* (Griewank, 2000). Then, some practical aspects regarding the use of AD will be addressed, particularly in conjunction with the simulation of a mechanical system involving the use of look-up tables, interpolating functions, step functions, etc., such as the mathematical model of a vehicle. The structure of the lap time optimisation program including AD will also be described. Finally, a simple minimum time vehicle manoeuvring problem will be set up and solved using both AD and numerical differentiation techniques, contrasting the relative computational performance and accuracy. I hope, too, that this will cause the reader a small chain rule reaction!

7.1. NUMERICAL DIFFERENTIATION AND LAP TIME OPTIMISATION

Let us consider a scalar function $f(q)$ of a single variable q. In order to evaluate its first order derivative, we may apply the definition of the derivative itself:

$$f'(q) = \lim_{h \to 0} \frac{f(q+h) - f(q)}{h} \qquad \text{Eq. 7.1}$$

Choosing a value of h sufficiently small, we may write:

$$f'(q) \cong \frac{f(q+h) - f(q)}{h} \qquad \text{Eq. 7.2}$$

Eq. 7.2, which is known as the forward difference formula, is the simplest procedure we can apply to compute derivatives. However, when applied blindly, this method is very likely to yield inaccurate results.

Derivatives obtained by numerical differentiation are affected by truncation and round off errors. The following analysis, reported in Numerical Recipes (Press et al., 1992), allows to find the optimal value for h which minimises the combination of such errors. Truncation errors come from the higher order terms of the Taylor series

expansion of the function $f(q)$:

$$f(q+h)=f(q)+h\cdot f'(q)+\frac{1}{2}\cdot h^2\cdot f''(q)+\frac{1}{6}\cdot h^3\cdot f'''(q)+\cdots \quad \text{Eq. 7.3}$$

and:

$$\frac{f(q+h)-f(q)}{h}=f'(q)+\frac{1}{2}\cdot h\cdot f''(q)+\cdots \quad \text{Eq. 7.4}$$

Hence, for the forward difference formula the truncation error will be of the order of:

$$e_t \sim \left|h\cdot f''(q)\right| \quad \text{Eq. 7.5}$$

Round off errors have various contributions which are related to the finite precision machine arithmetic. Assuming ε_f as the fractional accuracy with which the function f is evaluated, the round off error for Eq. 7.2 will be of the order of:

$$e_r \sim \varepsilon_f \left|f(q)/h\right| \quad \text{Eq. 7.6}$$

Then, the value for h which minimises the sum of truncation and round off errors given by Eqs. 7.5 and 7.6 respectively reads:

$$h\sim\sqrt{\frac{\varepsilon_f\cdot f}{f''}}=\sqrt{\varepsilon_f}\cdot\sqrt{\frac{f}{f''}}=\sqrt{\varepsilon_f}\cdot q_c \quad \text{Eq. 7.7}$$

Here, q_c is defined as the curvature scale of the function f. Often, no information is available regarding the curvature scale of the function, in which case one may assume $q_c \cong q$. Using this optimal value for h in Eqs. 7.5 and 7.6, the fractional accuracy of the derivatives computed with Eq. 7.2 may be estimated as follows:

$$\varepsilon_d=(e_t+e_r)/\left|f'\right|\sim\sqrt{\varepsilon_f} \quad \text{Eq. 7.8}$$

Therefore, the fractional accuracy of the computed derivatives will be at best of the order of the square root of the fractional accuracy for the function evaluation. If one may afford two function evaluations for each derivative, it is significantly better to use a centred difference formula:

$$f'(q)\cong\frac{f(q+h)-f(q-h)}{2h} \quad \text{Eq. 7.9}$$

In this case the truncation error becomes of the order of:

$$e_t \sim \left|h^2\cdot f'''(q)\right| \quad \text{Eq. 7.10}$$

while the round off error remains of the same order. By doing the same analysis as above, the optimal value of h for the centred difference formula may be derived and its expression reads:

$$h \sim \left(\frac{\varepsilon_f \cdot f}{f'''} \right)^{1/3} = \left(\varepsilon_f \right)^{1/3} \cdot q_c \qquad \text{Eq. 7.11}$$

When applying Eq. 7.9 with the optimal value for h returned by Eq. 7.11, at best the fractional accuracy of the derivatives will be of the order of:

$$\varepsilon_d \sim \left(\varepsilon_f \right)^{2/3} \qquad \text{Eq. 7.12}$$

Still, the derivatives are computed with a significant loss in precision compared to the function. Unfortunately, better accuracy may only be obtained at the cost of several function evaluations for each derivative, and this is not practical for large scale optimisation problems.

Even though this basic error analysis is a good starting point, the evaluation of the derivatives for solving the minimum lap time optimisation problem defined by Eq. 6.31 by means of numerical differentiation techniques requires some further considerations. First of all, the evaluation of the lap time and the various state constraints requires the integration of the vehicle dynamics differential equations. Therefore the fractional accuracy with which the objective and the constraint functions are evaluated will not be limited by the machine accuracy, as would be expected for algebraic functions, but will be limited by the accuracy with which the integration of the differential equations is performed, which is usually several orders of magnitude poorer compared to machine accuracy. Hence, it will be necessary to set tighter tolerances for the ODE integrator in order to compensate for the loss of accuracy in the derivatives due to the numerical differentiation procedure, therefore increasing the computational cost further.

When dealing with vector valued functions of many independent variables, such as in the present case, it is a good practice to scale the problem in order to make the sensitivities of the many dependent variables as close as possible to each other in magnitude. Then, Eq. 7.7 or Eq. 7.11 may be applied to determine a suitable value for h. However, a few trials should be done in order to validate such value. This may be accomplished by simply repeating the evaluation of the derivatives varying h within an arbitrarily large range and plotting a measure of the magnitude of the derivatives, e.g. the norm of the gradient of the objective function, against the values of h. Typically, for large values of h the derivatives will be mostly affected by truncation errors, while for small values of h round off errors will be predominant. Between these two extremes there ought to be an interval of values for h which yield consistent derivative results and there is where its optimal value should be found.

The parallel shooting algorithm which was proposed in chapter 6 for the solution of the minimum lap time optimisation problem implies a sparsity pattern which may be exploited in order to minimise the computational cost for evaluating the derivatives. In other words, since the vehicle trajectory is divided into completely de-coupled segments, we do not need to repeat the simulation of the entire lap for each derivative we want to evaluate by numerical differentiation. Instead, we only need to repeat the simulation of the segment of vehicle trajectory which changes as a result of the variation

of a particular independent control or state variable. As an example, let us refer to figure 7.1 and consider the track segment included between the path co-ordinates s_i and s_{i+1}. Here, we will have to find the sensitivity of the objective function, i.e. the lap time, and of the constraints with respect to the initial state vector (indicated with $\mathbf{x}_i$) and the control parameters which occur within that segment. Because of the de-coupling, only a sub-set of the total problem constraints will be affected by the variation of the independent variables named above. This sub-set includes the road boundary constraints which may occur in the segment, and the continuity state constraints with the neighbouring segments. Given the problem discretisation scheme, we are able to identify the minimum computational effort which is required for the numerical differentiation of problem 6.31. Firstly, when evaluating the sensitivity with respect to the initial state vector $\mathbf{x}_i$, we need to repeat the simulation of the entire track segment, i.e. from s_i to s_{i+1}, for each of the state parameters. Secondly, when evaluating the sensitivity with respect to the control parameters, we may observe that for a variation of a control parameter which occurs at a generic path co-ordinate s_j the overall control history will change only from the previous control node s_{j-1}, when assuming linear interpolation between the control points. Therefore, to find the sensitivity with respect to a control parameter located at s_j we only need to repeat the simulation of the trajectory from s_{j-1} to s_{i+1}. In order to be able to re-start the integration at any intermediate control point of the segment, we need to store the values of the nominal vehicle state variables, i.e. the states which correspond to the nominal control input, at each node of the communication grid. Despite the increased complexity involved, this strategy is essential in order to reduce the computational time to acceptable levels.

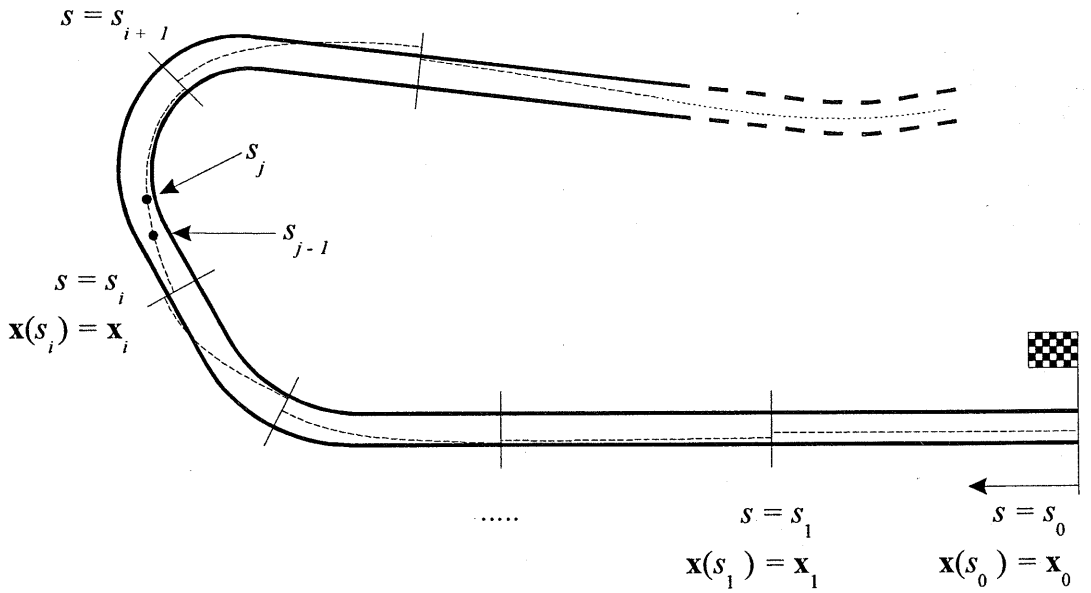


Figure 7.1 Exploiting sparsity for evaluating the derivatives.

7.2. AUTOMATIC DIFFERENTIATION TECHNIQUES BY EXAMPLES

In this section the basic forward and reverse accumulation techniques for AD are described by using a simple example. The simple theory behind the description which is given here may be found in (Bartholomew-Biggs et al., 1994). We will not consider how these techniques may actually be implemented in a computer program, as this is beyond the scope of the work, even though a brief insight will be given at the end of the section. Here, we will focus on the sequence of operations that the computer must perform in order to evaluate the gradient of a function using AD techniques. For this purpose we shall refer to the following algebraic function of four independent variables:

$$y = f(x_1,\ x_2,\ x_3,\ x_4) = x_1 \cdot x_2 + x_3 \cdot \sin(x_4) \qquad \text{Eq. 7.13}$$

For this simple function, the symbolic expression of the gradient reads:

$$\nabla f = [x_2 \quad x_1 \quad \sin(x_4) \quad x_3 \cdot \cos(x_4)] \qquad \text{Eq. 7.14}$$

7.2.1. Gradient by forward accumulation

With the forward accumulation technique it is firstly necessary to associate to each program variable a vector which carries the partial derivatives of that variable with respect to the independent ones. For this purpose, let us define a new data type, which we shall call *doublet*, as follows:

$$doublet \quad X = (x, x')$$

The first item of the *doublet* is a scalar and is the actual value of the variable, while the second item is a vector which contains its partial derivatives.

Next, we may observe that the evaluation of a function consists of a sequence of elementary operations, such as arithmetic operators (e.g. add, multiply, etc.) and intrinsic functions (e.g. sine, cosine, etc.). Let us then assume that we may somehow re-define these operators and functions in such a way that when they apply to *doublets*, the corresponding derivative operations are performed. For our example of Eq. 7.13 we will have two arithmetic operators and one function, which would have to perform the following operations when applied to *doublets*:

$$\begin{aligned} X \cdot Y &= (x \cdot y,\ x' \cdot y + x \cdot y') \\ X + Y &= (x + y,\ x' + y') \\ \sin(X) &= (\sin(x),\ x' \cdot \cos(x)) \end{aligned} \qquad \text{Eq. 7.15}$$

In other words, we program the computer in such a way that the actual algebraic operations are performed on the scalar part of the *doublets*, while the corresponding derivatives are computed on the vector part. The re-definition of these operators is the only stage which is similar to symbolic differentiation, since we actually program the differentiation rule corresponding to each operator. When executing the function, though, such operations will be executed on numerical values only.

We are now nearly ready to evaluate our function using AD in the forward

accumulation mode. What is left to do is to declare our program variables as *doublets* and initialise them properly. For the independent variables we may write:

doublet X_1, X_2, X_3, X_4

$$X_1 = \left(x_1, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right); \quad X_2 = \left(x_2, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \right); \quad X_3 = \left(x_3, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \right); \quad X_4 = \left(x_4, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right);$$

Then, for the dependent variable we have:

doublet Y

$$Y = \left(y, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right);$$

The computer program for evaluating the function defined by Eq. 7.13 may now be written exactly in the same way as if we were not applying AD:

$$Y = X_1 \cdot X_2 + X_3 \cdot \sin(X_4)$$

As the program is executed, though, the computer will perform the derivative operations which have been defined and apply the chain rule automatically. Using the definitions given above for the *doublets* and for the three operators in Eq. 7.15, we may view the sequence of algebraic operations which are performed and show how the derivatives are propagated into the vector part of the *doublets*:

1) Firstly, execute the sin() operator:

$$Y = X_1 \cdot X_2 + X_3 \cdot \left(\sin(x_4), \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot \cos(x_4) \right)$$

$$Y = X_1 \cdot X_2 + X_3 \cdot \left(\sin(x_4), \begin{bmatrix} 0 \\ 0 \\ 0 \\ \cos(x_4) \end{bmatrix} \right)$$

2) Then, execute the two multiplications:

$$Y = \left(x_1 \cdot x_2, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot x_2 + x_1 \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \right) + \left(x_3 \cdot \sin(x_4), \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \cdot \sin(x_4) + x_3 \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ \cos(x_4) \end{bmatrix} \right)$$

$$Y = \left(x_1 \cdot x_2, \begin{bmatrix} x_2 \\ x_1 \\ 0 \\ 0 \end{bmatrix} \right) + \left(x_3 \cdot \sin(x_4), \begin{bmatrix} 0 \\ 0 \\ \sin(x_4) \\ x_3 \cdot \cos(x_4) \end{bmatrix} \right)$$

3) Finally, execute the addition:

$$Y = \left(x_1 \cdot x_2 + x_3 \cdot \sin(x_4), \begin{bmatrix} x_2 \\ x_1 \\ \sin(x_4) \\ x_3 \cdot \cos(x_4) \end{bmatrix} \right)$$

Hence, at the end of the computation the *doublet Y* will contain the numerical values of the function and its gradient.

This simple forward accumulation technique allows to obtain derivatives with the same accuracy as the function values. However, it is not very efficient and this is clear from the computation described above, where many algebraic operations between the vector parts of the *doublets* involve quantities which are actually equal to zero.

7.2.2. Gradient by reverse accumulation

The reverse accumulation technique for AD allows to avoid algebraic operations between quantities which are equal to zero by keeping trace of the mutual dependencies among the program variables. This is achieved by representing a function by means of its *computational graph.*

The evaluation of a function defined by a computer program involves a sequence of unary operators, i.e. operators which involve only one program variable such as $\sin(x)$, and binary operators, i.e. operators which involve two program variables such as $x \times y$. The result yielded by each operator is not assigned to any of the variables which are explicitly declared in the program, but it will reside in the computer memory as long as it is needed. To illustrate the procedure, let us refer to the function defined by Eq. 7.13. In evaluating such function, the computer performs the following steps:

- Evaluate $\sin(x_4)$ and store the result in the memory;
- Take the previous result, multiply it by x_3 and store the result in the memory;
- Multiply x_1 by x_2 and store the result in the memory;
- Finally, take the previous two results, add them together and return the function value.

The computational graph may be described as the recording of such sequence of operations. It essentially consists of a sequence of *nodes*. Each node stores one of the values assumed by the many program variables involved in the computation. The first n nodes usually store the independent variables, while the subsequent nodes store the results yielded by the unary and binary operators involved. Formally, for a generic scalar function $y = f(x_1,\ldots,x_n)$, with n independent variables, the nodes of the computational graph may be defined as follows:

$$x_i = f_i(x_1,\ldots,x_{i-1}) \qquad i = n+1,\ldots,m \qquad \text{Eq. 7.16}$$

For generality, each node defined by Eq. 7.16 is written as dependent on all the previous nodes. However, since each node involves either one unary or one binary operator in the computational graph of f, it will only be a function of one or two of the previous nodes. From the definition given above, it follows that the last node of the computational graph holds the function value:

$$y = x_m \qquad \text{Eq. 7.17}$$

Next, the evaluation of the function derivatives using the reverse accumulation method relies on the *adjoint* variables (in fact, the reverse accumulation is often referred to as the adjoint method). Each node of the computational graph is associated to one adjoint variable, which is defined as the partial derivative of the function value y with respect to that node. The formal definition for the adjoints reads:

$$\bar{x}_i = \frac{\partial y}{\partial x_i} = \frac{\partial x_m}{\partial x_i} \qquad i = 1,\ldots,m \qquad \text{Eq. 7.18}$$

Consequently, the last adjoint variable reads the following value:

$$\bar{x}_m = \frac{\partial x_m}{\partial x_m} = 1 \qquad \text{Eq. 7.19}$$

Clearly, from the definition given in Eq. 7.18 it follows that the adjoint variables for the first n nodes of the computational graph will hold the values of the gradient of the function. The evaluation of the adjoint may be accomplished in reverse order, starting from $\bar{x}_m$ whose value is known, by applying the chain rule:

$$\bar{x}_j = \sum_{i>j} \bar{x}_i \cdot \frac{\partial f_i}{\partial x_j} \qquad j < m \qquad \text{Eq. 7.20}$$

Formally, we may expand Eq. 7.20 and define the reverse pass as a loop as follows:

for $i = m$ *down to* $n+1$

$$\bar{x}_j = \bar{x}_j + \frac{\partial f_i}{\partial x_j} \cdot \bar{x}_i \qquad j = 1,\ldots,i-1 \qquad \text{Eq. 7.21}$$

end

And, finally:

$$\nabla f = \bar{x}_i \qquad i = 1,\ldots,n \qquad \text{Eq. 7.22}$$

Let us now apply this technique to our function defined by Eq. 7.13 and see explicitly what operations are involved. Firstly, the computational graph for that function reads:

$$\begin{aligned}
x_1 &= x_1 \\
x_2 &= x_2 \\
x_3 &= x_3 \\
x_4 &= x_4 \qquad\qquad n = 4 \\
x_5 &= f_5(x_1, x_2) = x_1 \cdot x_2 \\
x_6 &= f_6(x_4) = \sin(x_4) \\
x_7 &= f_7(x_3, x_6) = x_3 \cdot x_6 \\
x_8 &= f_8(x_5, x_7) = x_5 + x_7 \qquad m = 8
\end{aligned}$$

In total, we have $n = 4$ independent variables and $m = 8$ nodes. Hence, we will have $\bar{x}_8 = 1$. Next, we may execute the reverse pass by performing the steps stated in Eq. 7.21, bearing in mind that only one or two of the partial derivatives of each node f_i will be non-zero:

1) $i = 8$. The node f_8 depends on nodes number 5 and 7. Hence:

$$\bar{x}_5 = \bar{x}_5 + \frac{\partial f_8}{\partial x_5} \cdot \bar{x}_8 = 1 \cdot \bar{x}_8 = 1$$

$$\bar{x}_7 = \bar{x}_7 + \frac{\partial f_8}{\partial x_7} \cdot \bar{x}_8 = 1 \cdot \bar{x}_8 = 1$$

2) $i = 7$. The node f_7 depends on nodes number 3 and 6. Hence:

$$\bar{x}_3 = \bar{x}_3 + \frac{\partial f_7}{\partial x_3} \cdot \bar{x}_7 = x_6 \cdot 1 = x_6 = \sin(x_4)$$

$$\bar{x}_6 = \bar{x}_6 + \frac{\partial f_7}{\partial x_6} \cdot \bar{x}_7 = x_3 \cdot 1 = x_3$$

3) $i = 6$. The node f_6 depends only on node number 4. Hence:

$$\bar{x}_4 = \bar{x}_4 + \frac{\partial f_6}{\partial x_4} \cdot \bar{x}_6 = \cos(x_4) \cdot x_3$$

4) $i = 5$. The node f_5 depends on nodes number 1 and 2. Hence:

$$\bar{x}_1 = \bar{x}_1 + \frac{\partial f_5}{\partial x_1} \cdot \bar{x}_5 = x_2 \cdot 1 = x_2$$

$$\bar{x}_2 = \bar{x}_2 + \frac{\partial f_5}{\partial x_2} \cdot \bar{x}_5 = x_1 \cdot 1 = x_1$$

Finally, here are the adjoint variables which hold the components of the gradient of f:

$$\nabla f = [\bar{x}_1 \quad \bar{x}_2 \quad \bar{x}_3 \quad \bar{x}_4] = [x_2 \quad x_1 \quad \sin(x_4) \quad x_3 \cdot \cos(x_4)]$$

Compared to the previous example regarding the forward accumulation technique, here all the algebraic operations involving 0s have been avoided. This follows from the definition of the nodes, which only involve one or two program variables at a time. Obviously, the programming strategy to keep trace of the mutual dependencies among the program variables is not simple, but such strategy is the strength of the reverse accumulation technique.

7.2.3. Implementation strategies for Automatic Differentiation

So far, an overview of the basic techniques for AD has been presented without referring to any strategy for actually implementing such techniques into a computer program. We will now make a final step and give a brief insight on how this is achieved by the existing tools for Automatic Differentiation. AD tools work by modifying the semantics of the underlying program by inserting, in a rule-based fashion, code for computing derivatives. There are two main approaches to achieve this and they are described as follows.

- **Object oriented approach by means of operator overloading:** This approach overloads the basic arithmetic operators and intrinsic functions with special routines that carry out the propagation of derivatives in addition to the original operations, as was described, for example, with regard to the forward accumulation mode. Alternatively, the necessary information for the code reversal may be recorded at each node of the computational graph in the reverse accumulation method. The source program itself is only minimally changed, and most of the complexity of derivative computations is embedded in a library. The operator overloading mechanism properly invokes these library routines at run time, as the execution proceeds. This approach is applicable with programming languages that support operator overloading such as C++ or Fortran 90.
- **Source-to-source transformation:** This approach employs compiler techniques to transform a program source code into a new source code that explicitly carries out the derivative computation. Hence it is applicable to any language. This approach is less elegant and more difficult to implement compared to the previous one. However

it may be more efficient since the entire program context is available at compile time and therefore there are more possibilities for optimisation. Furthermore, the mechanism of operator overloading itself involves some overheads which penalise the efficiency of the object oriented approach.

Various tools for Automatic Differentiation have been implemented using both approaches and different programming languages. A list may be found at the following web site: `http://www.mcs.anl.gov/Projects/autodiff/AD_Tools`. Usually, both forward and reverse techniques are implemented and the optimal strategy is automatically applied depending on the structure of the function which is to be differentiated. In (Bartholomew-Biggs et al., 1994) various techniques are compared for solving different optimisation problems. Finally, the combination of the reverse accumulation technique and *doublet* arithmetic allows the evaluation of higher order derivatives (Christianson, 1992). Although this is very important, as it opens the possibility to apply quadratic methods in optimisation problems which ensure faster convergence, we will not go any further here. Next, we will focus on how AD may be integrated with the lap time optimisation program.

7.3. AUTOMATIC DIFFERENTIATION AND LAP TIME OPTIMISATION

In this work an Automatic Differentiation software tool called **AD***opt* (= **A**utomatic **D**ifferentiation and *opt*imisation) is employed to evaluate the first order derivatives for the objective function and the many problem constraints for the lap time optimisation problem defined by Eq. 6.31, so that the problem may be solved using a Sequential Quadratic Programming algorithm. **AD***opt* is the result of a research program undertaken at the Numerical Optimisation Centre[1], University of Hertfordshire, Hatfield, by Mark Final and Prof. Bruce Christianson. Their expertise and their support has contributed a great deal to the implementation of AD in the lap time optimisation program.

AD*opt* is an object oriented, interpretative AD tool written in the C++ programming language. Both forward and reverse schemes are implemented to differentiate scalar or vector-valued functions. The set of operators and functions which are incorporated in **AD***opt* is shown in Table 7.1. Three specific functions for second (`_square()`), third (`_cube()`) and fourth (`_quad()`) powers have been defined, which guarantee greater efficiency for these particular cases compared to the standard power function (`pow()`). Finally, a special step function has been added to handle specific problems, e.g. the modelling of the gear shift, as we shall see. The step function is defined as follows:

$$step(x, x_0, a, b,) = \begin{cases} a & if \quad x < x_0 \\ \dfrac{a+b}{2} & if \quad x = x_0 \\ b & if \quad x > x_0 \end{cases} \qquad \text{Eq. 7.23}$$

[1] Contact: `http://www.feis.herts.ac.uk/home.asp?filename=/research/RC.asp`

=	+	sin()	pow()	_quad()
+=	- (unary)	cos()	log()	sign()
-=	-	tan()	exp()	step()
*=	*	atan()	square()	
/=	/	sqrt()	_cube()	

Table 7.1 Operators defined in **AD***opt*.

In order to facilitate input and output operations involving vectors and matrices, additional interfaces are provided in **AD***opt*. These are the `RealVector` and the `RealMatrix` classes respectively. Specific functions allow to manipulate these vector and matrix objects, for example to extract rows from Jacobian matrices or elements from gradient vectors. Particularly, these functions aid the use of AD within constrained optimisation problems. Finally, the derivatives available from **AD***opt* are sufficient for unconstrained optimisation methods, such as Truncated Newton, as well as constrained optimisation methods such as the SQP algorithm which is used to solve the minimum lap time problem.

There are limitations to what can be done using **AD***opt*. Those which mostly affect its application to the lap time optimisation problem, as will become clear later on, are:

1. The computational graph is made of a single branch. Multiple branches, i.e. those which would arise out of conditional statements, are not possible.
2. Active loops are not allowed, i.e. "`for`" loops which iterate a number of times depending on the values assumed by a variable which is active in the computational graph.
3. Loops are completely unrolled onto the computational graph, i.e. the sequence of operations in the loop is recorded in the computational graph as many times as the loop is repeated.
4. Indices of active array variables can not be expressions of other active variables.

The aim of this section is to describe the application of AD to a computer program from a practical point of view. Even though we shall use the specific syntax defined in **AD***opt*, still the approach is sufficiently general in order to give an idea of what is involved in the program transformation when using an object oriented AD tool. Firstly, a general example will be described by referring once more to the function defined by Eq. 7.13. Then, we will focus on the lap time optimisation program and the specific problems which are involved in the evaluation of the objective and constraint functions through the simulation of the vehicle dynamics equations.

7.3.1. Program transformation with AD*opt*: the basic concepts

To begin with a simple example, let us write a program to evaluate the function defined by Eq. 7.13 using the C/C++ programming language (Kernighan and Ritchie, 1988, Stroustrup, 1997, Schildt, 1998). The layout of such program is described in figure 7.2.

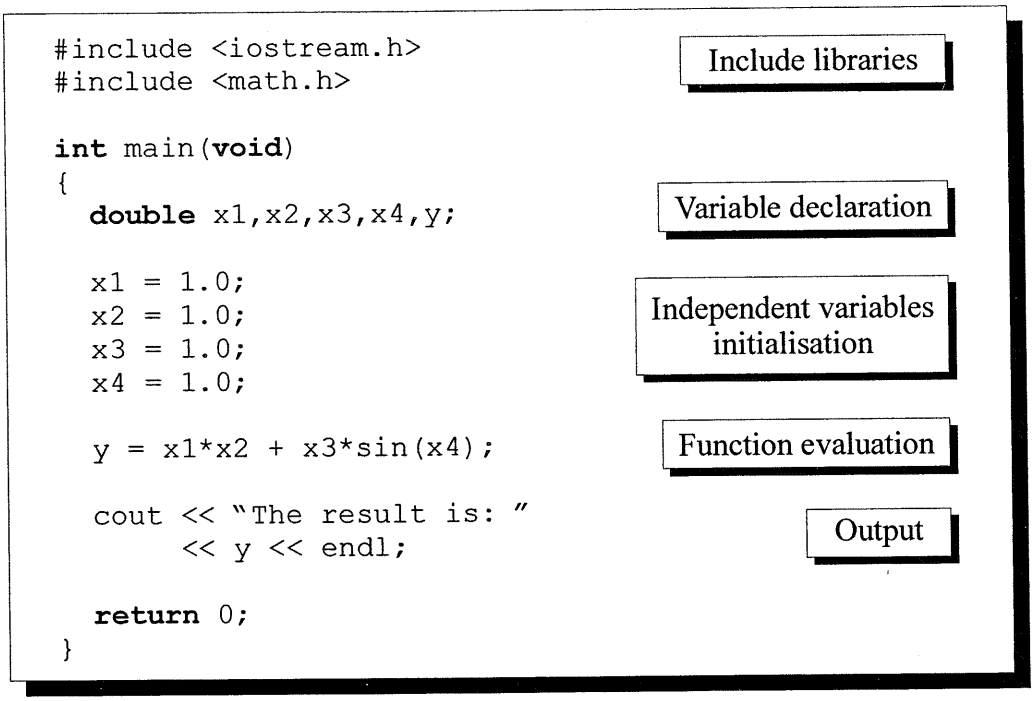


Figure 7.2 The structure of the computer program without AD.

Basically, the program includes the following sections:

- Include libraries: these lines tell the compiler to include the required libraries of functions, in this case the C++ input/output stream library and the math library.
- Variable declaration: define the name and the type of the variables involved in the computation.
- Initialisation of the independent variables: assign numerical values to the independent variables.
- Function evaluation.
- Output the results.

Figure 7.3 shows the computer program for the evaluation of the same function after the transformation for using **AD***opt* to obtain also its gradient. The steps involved in the transformation are described below.

- Include the "`ADopt.h`" library. This makes all the functions and utilities defined in the **AD***opt* package available in the program.
- Declare the `CodeList`. This is the data object which will hold the computational graph of the function, according to the nomenclature defined in **AD***opt*. Other AD packages, for example, refer to the same object as the "tape".
- Define the **AD***opt* scoping unit. In general, all the AD tools require the user to identify the active section of the code. That is, the part of the code where the actual computation of the chosen dependent variable(s) from the independent variable(s) takes place. In **AD***opt* the active part of the code is included in a pair of braces, which are the standard C/C++ notation to identify a scope of code. Furthermore, **AD***opt* requires the two functions `BeginComposition()` and

`EndComposition()` at the start and at the end of the composition of the computational graph respectively. These are essentially needed to inform **AD***opt* which one is its scoping unit.

```
#include <iostream.h>
#include <math.h>
#include "ADopt.h"

int main(void)
{
  CodeList myCL;
  {
    ADvar::BeginComposition();

    ADvar x1(&myCL,independent,1.0);
    ADvar x2(&myCL,independent,1.0);
    ADvar x3(&myCL,independent,1.0);
    ADvar x4(&myCL,independent,1.0);
    ADvar y(&myCL,dependent);

    y = x1*x2 + x3*sin(x4);

    ADvar::EndComposition();
  }

  myCL.CodeList__PrepareForAD();

  cout << "f = "
       << CodeList__GetFunctionValue(myCL)
       << endl;
  cout << "grad_f = "
       << CodeList__GetGradient(myCL)
       << endl;

  return 0;
}
```

Include libraries

Declare CodeList object

AD*opt* scoping unit

Active variables initialisation

Function evaluation

Prepare the CodeList

Extract results

Figure 7.3 The computer program transformed for AD.

- Declare and initialise the active variables. When applying AD to a general computer program, the user must identify the active variables. But which are these active variables? A rule to identify which variables are active (in the AD sense), besides the dependent and independent ones, may be the following. The computational graph may be thought of as a long chain of algebraic operations linking the dependent variable(s) to the independent variable(s). Along such a chain, many other variables may appear. Now, if we imagine to vary the independent(s), the program variables which vary too as a result, will be active. The AD tool must know which variables are active because it will need to take into account their partial derivatives when the chain rule is applied to the whole differentiation problem. In an

object-oriented AD tool the active variables are declared as a new data type, which in **AD***opt* is called `Advar`. Then, a further step must be taken in order to specify which of the active variables are the independents and which are the dependents. Actually, three types of `ADvar` active variables are specified in **AD***opt*, and these are:

- **Independent**, those variables which **AD***opt* calculates the derivatives with respect to.
- **Intermediate**, those variables which are active in the AD sense, but for which no derivatives are required.
- **Dependent**, those variables whose derivatives are required

As is shown in figure 7.3, in **AD***opt* it is possible to declare and initialise the active variables with a single command whose syntax is:

```
ADvar <name>(<CodeList address> <ADvar type> <value>)
```

This line of code contains the necessary information, such as the name of the variable, the memory address of the `CodeList` object to which it belongs (since there may be more than one `CodeList`, even though this possibility will not be used here), the type of active variable and its numerical value. Note that the values for dependent and intermediate variables are set equal to zero by default.

- Function evaluation. This part of the code remains unchanged.
- Preparing the `CodeList` for AD. Essentially this involves setting up the storage required by the forward and reverse sweeps of the `CodeList` which are necessary to extract the results.
- Extraction of the results. This is accomplished by calling some of the utilities defined in **AD***opt*. These utilities trigger the execution of the forward and reverse sweeps on the computational graph as needed. This is hidden from the user, as **AD***opt* chooses the most efficient option for the order of derivatives required. The results will be `RealVector` and `RealMatrix` objects.

7.3.2. Program transformation when calling user defined functions

Long computer programs for the evaluation of complex mathematical models are better written using functions. For example, a program for the simulation of a vehicle would consist of the main program scope which would call a function for the integration of the state differential equations, which in turn would call a function to evaluate the state derivatives as functions of the system states, which in turn would call other functions for the evaluation of forces, moments, etc. To illustrate how this program structure may be employed easily also when using AD, the initial example described in figure 7.2 has been modified by adding a user defined function to perform the actual evaluation of our simple problem. The new program is shown in figure 7.4.

First of all, in this new version of our example program the independent variables are declared as a four element array, rather than having four scalar variables. Then, the user defined function is declared in such a way that it does not return any value. Instead, the reference operator "`&`" is used for the function output "`y`". The reference operator is

available only in C++ and is equivalent to using a pointer for input and output of scalar function arguments. This means that the variable "y" inside the body of the function is treated as an alias of the actual argument, thus avoiding making a new copy of the same argument. Not only is this normally advantageous in terms of program efficiency, but is also a requirement for using **AD***opt*, as we shall see in a moment.

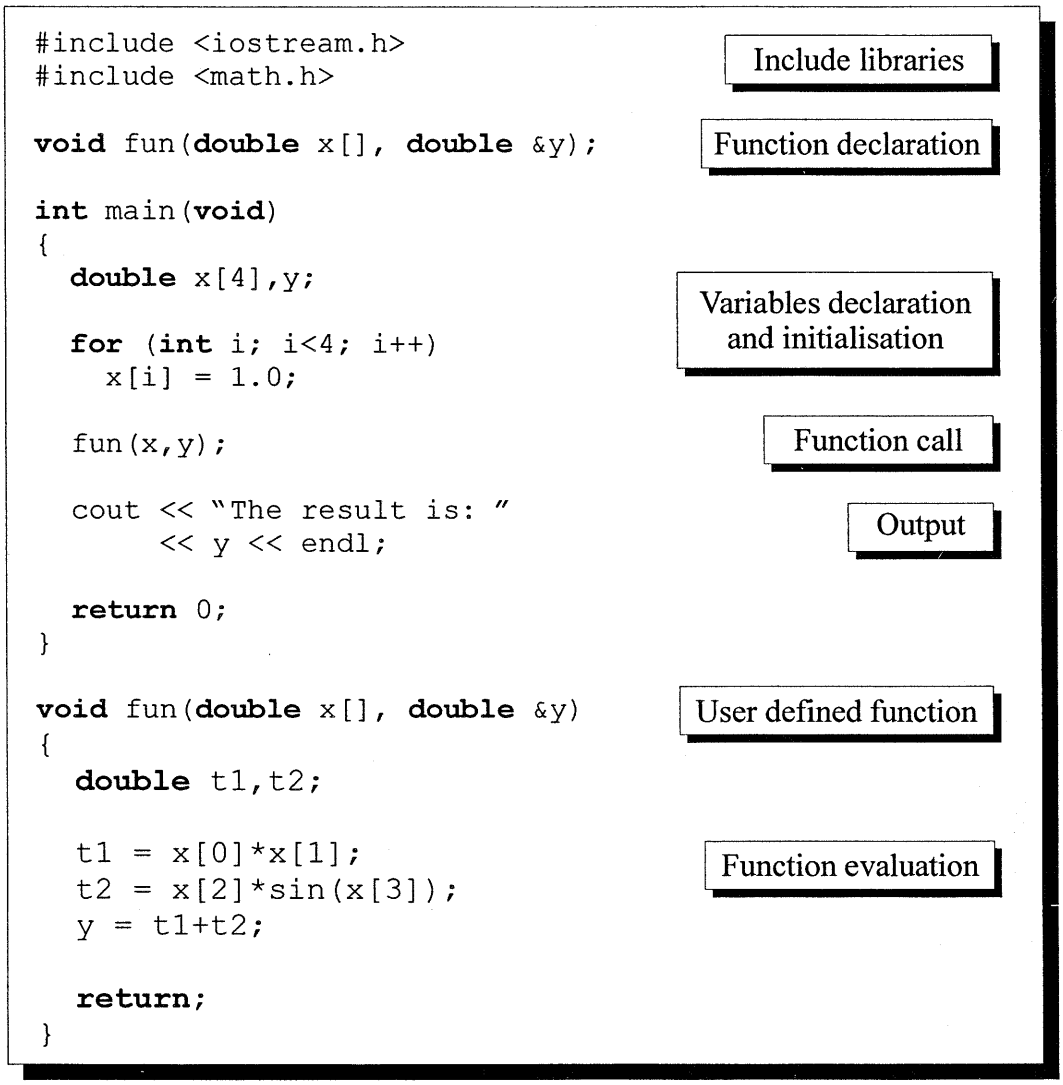


Figure 7.4 The computer program with user defined functions.

Figure 7.5 shows the new computer program with user defined functions transformed for using **AD***opt*. The necessary extra changes, in comparison with the previous implementation shown in figure 7.3, are:

- Function declaration. The user defined function is declared using the `ADvar` data type for the input and output arguments. Here, the use of the reference operator for the scalar output variable "y" (or, alternatively, the usual pointer notation which is common for C and C++) is necessary when using **AD***opt*. Passing the actual argument to the function would imply that **AD***opt* makes a copy of the same variable in the `CodeList` and this would compromise the AD procedure. Finally, if

local active variables are declared inside the body of the function (`t1` and `t2` in this case), it is necessary to pass to the function the memory address of the `CodeList` object to which they belong.

```
#include <iostream.h>
#include <math.h>
#include "ADopt.h"

void fun(ADvar x[],ADvar &y,CodeList *CL);

int main(void)
{
  CodeList myCL;
  {
    ADvar::BeginComposition();

    ADvar x[4], y(&myCL,dependent);
    for (int i; i<4; i++)
      x[i].ADvar__set(&myCL,independent,1.0);

    fun(x,y,&myCL);

    ADvar::EndComposition();
  }
  myCL.CodeList__PrepareForAD();

  cout << "f = "
       << CodeList__GetFunctionValue(myCL)
       << endl;
  cout << "grad_f = "
       << CodeList__GetGradient(myCL)
       << endl;

  return 0;
}

void fun(ADvar x[], ADvar &y, CodeList *CL)
{
  ADvar t1(CL,intermediate), t2(CL,intermediate);

  t1 = x[0]*x[1];
  t2 = x[2]*sin(x[3]);
  y = t1+t2;

  return;
}
```

Include libraries

Function declaration

Declare CodeList object

AD*opt* scoping unit

Active variables initialisation

Function call

Prepare the CodeList

Extract results

User defined function

Function evaluation

Figure 7.5 The computer program with user defined functions transformed for AD.

- `ADvar` array declaration. In this example we show also how arrays of type `ADvar` may be declared and initialised using `ADvar__set`.
- User defined function. The actual problem equation is written in the user defined function, where two intermediate active variables are also declared.

In conclusion, the changes regard essentially the declaration and initialisation of the variables, but the code which actually computes the function remains unchanged and this is the main strength of the object oriented approach.

7.3.3. Using AD*opt* in general constrained optimisation problems

When implementing a program for the solution of a constrained optimisation problem, the user needs to define a routine for the evaluation of the objective and the constraints as functions of the independent optimisation variables. Any time the optimisation algorithm returns a new (improved) set of optimisation variables, it calls this routine to obtain new objective and constraint values. Furthermore, unless the gradient of the objective and the Jacobian of the constraints are supplied by the user by some means, such as AD, the optimisation algorithm will evaluate the derivatives by using numerical differentiation techniques, yet calling the user defined routine several times.

If AD is used for the evaluation of the derivatives that the optimisation algorithm needs (and particularly an object-oriented AD tool such as **AD***opt*) the ideal strategy is to record the entire evaluation procedure of the objective function and of the constraints on the computational graph. The computational graph must then remain stored in the computer memory, unchanged, for the whole optimisation process. Then, what the user defined routine has to do is simply to update the values of the independent active variables and extract new results from the computational graph, i.e. function values and derivatives, as needed by the optimisation algorithm.

In figure 7.6 this ideal layout for an optimisation program using **AD***opt* is described. The first phase of the program involves the composition of the `CodeList`, which is done essentially using the programming techniques described above. Then the user defined objective and constraint function simply acts as the interface between the `CodeList` and the optimisation algorithm. At the start of the optimisation, the user must supply the initial guess for the solution, i.e. $\mathbf{x}_0$. The optimisation algorithm calls the objective and constraint functions passing such initial guess to them. Next, the independent active variables $\mathbf{x}$ in the `CodeList` are set equal to $\mathbf{x}_0$ and the objective and constraints, and their derivatives are evaluated for the first time. With this information the optimisation algorithm may start iterating, until the solution meets certain tolerance criteria.

7.3.4. Using AD*opt* in the lap time optimisation program

The evaluation of the objective and the problem constraints for the lap time optimisation program involves the numerical solution of the vehicle dynamics equations. After the general introduction given above on program transformation for using AD, we are nearly ready to describe how the objective and constraint functions for the lap time optimisation program may be organised. However, there are still a few problems regarding the application of AD which deserve some attention, and these are how to handle look-up tables and discontinuities, such as the gear shifts. Hence, we shall firstly

describe the strategy which is applied in these cases, showing how AD is still easily applicable.

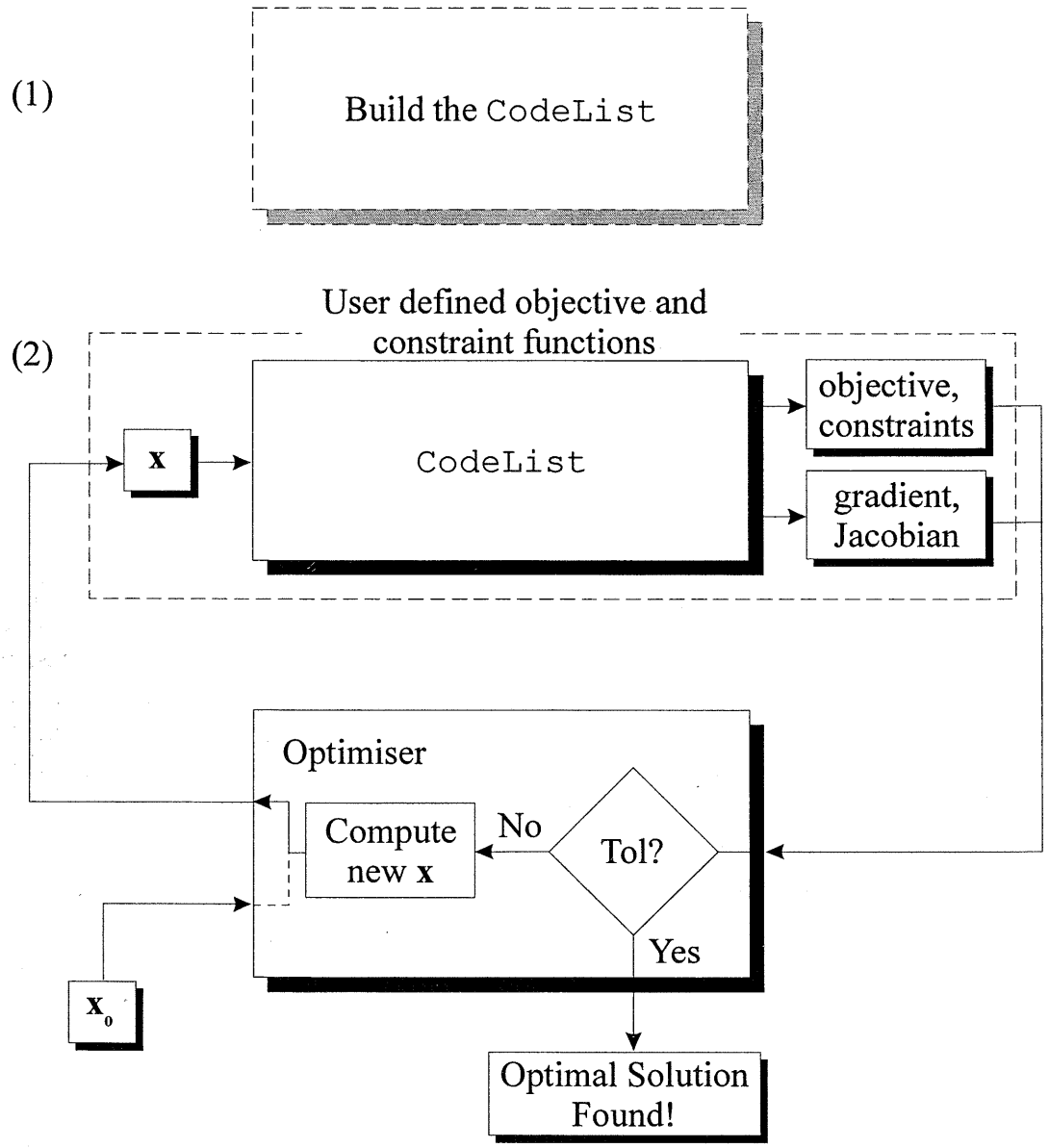


Figure 7.6 Structure of the optimisation program with AD.

7.3.4.1. Using look-up tables

For simplicity, let us consider a one-dimensional look-up table. Let $\mathbf{x}_n$ and $\mathbf{y}_n$ be the two vectors of n elements representing the input and output tabulated values respectively. The task is to estimate the output value y for an arbitrary input value x by means of linear interpolation:

$$y = \text{lin\_interp}(\mathbf{x}_n, \mathbf{y}_n, x) \qquad \text{Eq. 7.24}$$

Firstly, we need to identify the indexes of the two elements in the vector $\mathbf{x}_n$ which include x. If we assume that the elements in the vector $\mathbf{x}_n$ are equally spaced and Δx is their interval, this may be accomplished with a simple algebraic operation:

$$ind = floor\left(\frac{x - \mathbf{x}_n(0)}{\Delta x}\right) \qquad \text{Eq. 7.25}$$

Here, $\mathbf{x}_n(0)$ is the first element in the input vector of the table. The function *floor*() returns the greatest integer smaller than its argument. Assuming that the first index for the elements of the look-up table is 0, Eq. 7.25 returns the index of the largest element in $\mathbf{x}_n$ which is smaller than x. This method is much faster compared to the case of unevenly spaced elements in the input vector for the table, which requires special search algorithms to locate the index.

Next, we may perform the interpolation as follows:

$$y = \mathbf{y}_n(ind) + \frac{\mathbf{y}_n(ind+1) - \mathbf{y}_n(ind)}{\Delta x} \cdot (x - \mathbf{x}_n(ind)) \qquad \text{Eq. 7.26}$$

The application of AD to the linear interpolation is in principle straightforward. In Eq. 7.26, x and y will be active variables and the partial derivative of the output with respect to the input will simply be the slope between two points in the table. However, one of the limitations of **AD***opt* implies that we are not allowed to define indexes for arrays as functions of active variables, which would be the case here for Eq. 7.25. The workaround for this problem, though, is simple and consists of extracting the actual value of the active variable and copying it into a passive one using the utility `valueof()` defined in **AD***opt*. Hence, we only need to modify Eq. 7.25 as follows:

$$ind = floor\left(\frac{valueof(x) - \mathbf{x}_n(0)}{\Delta x}\right) \qquad \text{Eq. 7.27}$$

Finally, the same framework may be extended to cover more complex cases such as *n*-dimensional look-up tables and the use of spline interpolation techniques.

7.3.4.2. Modelling the gear shifts

In a vehicle model for handling and performance analysis, where computational speed is essential, the gear shift may simply be represented as a step function of the vehicle velocity. The step function described by Eq. 7.23 was introduced in **AD***opt* mainly for this purpose. For example, if V is the vehicle velocity, V_{12} is the velocity at which the vehicle shifts from first to second gear and τ_1 and τ_2 are the first and second gear ratios, we may write:

$$\tau = \text{step}(V, V_{12}, \tau_1, \tau_2) \qquad \text{Eq. 7.28}$$

When V is smaller than V_{12} the step function returns the first gear ratio. Instead, when V is greater than V_{12} it yields the second gear ratio. If V is equal to V_{12}, though, the choice is arbitrary and in this case the step function returns the average of the two possible

outputs, see Eq. 7.23. The derivative of the step function with respect to its input value is equal to zero everywhere, except for the case when V is equal to V_1, where it is not defined. However, **AD***opt* allows to ignore this point and arbitrarily set the undefined derivative to zero as well. In fact, the gear shifts in the vehicle simulation occur instantaneously, hence this assumption has no effect on the results.

Alternative strategies to model the gear shifts involve the use of continuous functions to approximate the discontinuity. This approach may be advantageous for the accuracy of the simulation, even though it increases slightly the computational load. In this case, though, the application of AD does not present any problem.

7.3.4.3. Evaluation of the objective and constraint function for lap time optimisation

All the elements are now in place for programming the evaluation of the lap time optimisation objective and constraint functions using **AD***opt*. Figure 7.7 shows the ideal structure of the computational graph. Firstly, we have to identify the active variables and divide them into independent, dependent and intermediate. Referring to the algorithm described in §6.4, we will have:

- **Independent active variables**: these will include the vector **y** of the independent optimisation variables, defined in Eq. 6.26, which consists of:
  - The vector of control parameters $\mathbf{u}_n$;
  - The vector of state parameters $\mathbf{x}_m$;
- **Intermediate active variables**: these will include all of the following variables which are evaluated at each integration step of the vehicle simulation:
  - The actual control input **u**, see Eq. 6.22;
  - The vectors **x** and $\dot{\mathbf{x}}$, i.e. the vehicle states and state derivatives;
  - Kinematic quantities function of the states, such as the wheel slip quantities, the vehicle distance from road centre line, etc.;
  - Forces and moment, i.e. aerodynamic forces, tyre vertical loads, tyre contact forces, etc.;
- **Dependent active variables**: these will include the objective and the constraints, such as:
  - The lap time J;
  - The continuity constraints, ξ;
  - The road boundary constraints, **c**.

Next, the integration loops for each of the m trajectory segments must be recorded on the computational graph. Within each segment, the initial value problem stated in Eq. 6.27 is solved using a fixed step, second order Runge Kutta integration formula (midpoint method). Assuming Δs as the length of the integration step, one may write:

for $s_{j-1} \le s_i \le s_j$ *and* $j = 1,\ldots,m$

$$\mathbf{x}\left(s_i + \frac{\Delta s}{2}\right) = \mathbf{x}(s_i) + \frac{\Delta s}{2} \cdot \bar{\mathbf{a}}(\mathbf{x}(s_i), \mathbf{u}(s_i), s_i) \qquad \text{Eq. 7.29}$$

$$\mathbf{x}(s_i + \Delta s) = \mathbf{x}(s_i) + \Delta s \cdot \bar{\mathbf{a}}\left(\mathbf{x}\left(s_i + \frac{\Delta s}{2}\right), \mathbf{u}\left(s_i + \frac{\Delta s}{2}\right), s_i + \frac{\Delta s}{2}\right)$$

end

The computational graph so obtained may finally be implemented in the optimisation scheme which was described in figure 7.6. The optimisation algorithm would communicate with the computational graph by installing new vectors of optimisation variables at every iteration. In turn, using the utilities defined in **AD***opt*, function values and their derivatives may be obtained as needed.

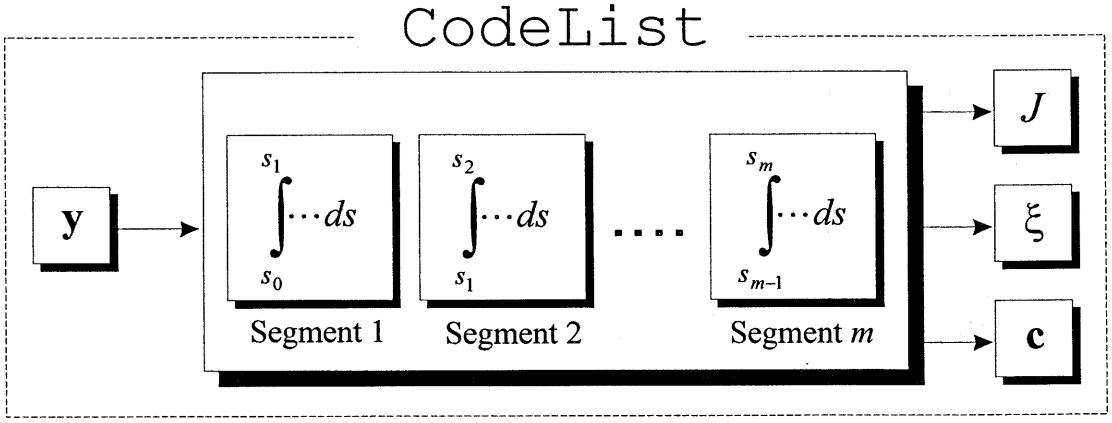


Figure 7.7 The Computational graph for the lap time optimisation objective and constraint function.

7.3.5. How the limitations in AD*opt* affect the lap time optimisation program

In summary, in this section the ideal strategy to use Automatic Differentiation within a general optimisation program, and particularly within the lap time optimisation program has been described. At this stage, though, we must take a step back. The limitations within **AD***opt*, which were highlighted earlier in this paragraph, prevent us from using the full potential of AD. First of all, in **AD***opt* each loop is unrolled onto the computational graph. This means that all the algebraic operations involved in each integration step are recorded as many times as the number of integration steps needed to complete a lap simulation. The storage of such a massive `CodeList` would require an amount of memory which is still not so common, neither on personal computers, nor on workstations. For a lap simulation with the vehicle model described in Appendix A, the memory required would be of the order of 5 to 10 Gbyte!

Secondly, even if we were able to solve the storage problem, still we would not be able to use the same `CodeList` for each iteration of the optimisation process. The problem lies in the way that interpolation techniques are handled using **AD***opt*. When

the index in a look-up table is evaluated using Eq. 7.27, its value is de-linked from the active variables, because indexes of arrays may not be functions of active variables. This is acceptable for the first time that we build and use the computational graph. But if we try to re-use it, such index will not be updated as it should be, as a consequence of the variation of the active variable which had generated it the first time. Hence we would obtain wrong results from the interpolation.

The way to overcome these problems is simple. The `CodeList` has to be built and destroyed for the simulation of each trajectory segment, as will be described more in detail in the next chapter. This, of course, will be less efficient than the ideal scheme presented here. It must be said that the only reason for having to choose a compromise was lack of time rather than lack of technology. At the end of this chapter we shall talk briefly about the technology which is needed to apply AD at its best. Still, though, the full potential of AD will be shown in the next paragraph where a problem, simpler than a full lap time optimisation, but sufficient to demonstrate the effectiveness of AD, is solved.

7.4. THE MINIMUM TIME VEHICLE MANOEUVRING PROBLEM: AUTOMATIC DIFFERENTIATION VS. NUMERICAL DIFFERENTIATION

In order to demonstrate the computational performance advantage offered by AD in comparison with numerical differentiation techniques, a simple minimum time vehicle manoeuvring problems has been set up, which allows to overcome the limitations imposed by **AD***opt*. This problem has been solved firstly supplying the derivatives to the optimisation algorithm using AD, and then letting the optimisation algorithm work out those derivatives by means of numerical differentiation. A different approach to compare the computational performance of these two differentiation methods could have simply been to show the different computational times for obtaining derivatives. However, a better picture arises when looking at how the optimisation program behaves in the two different situations. As we shall see in a moment, not only does AD allow to gain considerable computational speed. Also, all other things being equal, it allows the optimisation to converge to tighter tolerances.

The optimisation problem which is solved here involves a single corner manoeuvre. The length of the manoeuvre is limited to 300 m, so that a `CodeList` of reasonable size, i.e. about 300 Mb, is obtained. Furthermore, the vehicle model is simplified compared to the standard model which is currently implemented in the lap time optimisation program, presented in the next sections. Particularly, the look-up tables which could not be handled properly have been substituted with polynomial interpolating functions. A brief description of the vehicle model is given next.

7.4.1. The simplified vehicle model

In the optimisation problem presented here, the vehicle is represented with a five degrees of freedom model. The chassis is described as a rigid body with three degrees of freedom, the yaw angle and the lateral and longitudinal displacements. The wheels are rigidly attached to the body, allowing only the rotation relative to the vehicle chassis about their spin axes. Furthermore, the wheels on the same axle are constrained to have the same angular velocity. Therefore only two extra degrees of freedom are added.

Although the suspensions are not included in the modelling, the roll axis position, the roll stiffness distribution, the mass centre height and the track width are parameters taken into account in order to allow the evaluation of realistic wheel loads. The geometry of the steering system is not included in the modelling and the steer angle is thought to be the same for the left hand side wheel and the right hand side wheel.

A simple representation of the aerodynamic forces is employed by assuming constant drag and lift coefficients. The aerodynamic drag is applied at the height of the vehicle centre of gravity, so that it gives a front to rear load transfer in addition to that due to the aerodynamic lift and pitching moment. The centre of application of the aerodynamic lift is the same for all speeds and is determined by specifying the down force distribution between the front and the rear axles.

The tyre lateral and longitudinal forces are introduced using the Magic Formula Tyre Model for combined slip conditions (Pacejka and Besselink, 1997). The tyre slip quantities are evaluated with the assumption of small angles and they are referred to the centre of the front and rear axles, as for a bicycle model. Static wheel camber angle settings are accounted for. The wheel vertical loads are evaluated accounting for the vehicle static weight distribution, the aerodynamic down-force distribution and the longitudinal and lateral load transfers when the vehicle is driving, braking and cornering. The load transfers are estimated on the basis of a steady state analysis considering the vehicle longitudinal and lateral accelerations.

The longitudinal control variable is the throttle and brake input. It may assume values between –1 and +1. When positive, the engine output torque is applied to the rear wheels through a global gear ratio. The engine torque output, though, is not modelled here using a two dimensional look-up table (function of throttle aperture and engine rotational speed), as is the case for the complete model described in Appendix A. Instead, the maximum and minimum (engine braking effect) torque outputs from the engine are approximated with two polynomial functions of the engine rotational velocity only. At each integration step, the two maximum and minimum possible engine outputs, which correspond to the values of 0 and +1 for the control input, are computed. Then, using the actual value for the longitudinal control variable, a simple linear interpolation yields the torque applied to the rear wheels.

Finally, when braking, the longitudinal control variable simply factors the maximum braking torque available. Then the actual braking torque is shared among the front and rear axles using constant coefficients. The engine brake effect is also added to the rear axle.

7.4.2. Setting up the optimisation problem

Figure 7.8 shows the initial and optimal vehicle trajectories and controls for the manoeuvre. The road section is 300 m long. The control interval has been chosen to be equal to 12 [m] for both steer angle and throttle/brake controls. Thus, the problem has 50 independent variables in total (two control variables for 25 control nodes). For this short problem, it has not been necessary to divide the trajectory in shorter segments to achieve a robust behaviour. The objective function is the time it takes for the vehicle to traverse the road section. Finally, there are three road boundary constraints which are located where the optimal trajectory is tangent to the road boundaries. The vehicle starts from a fixed position at an initial velocity of 80 m/s and no other state constraints are applied.

Initially, a convergence test is performed in order to identify the integration step which ensures sufficient accuracy for the vehicle simulation. In this case, though, the convergence of the objective and constraint functions does not automatically guarantee the convergence of the derivatives obtained by AD. Therefore we need to monitor the value of the derivatives and ensure that consistent values are obtained as well. Figure 7.9 shows the initial objective value (corresponding to the initial guess of the control history) and its corresponding directional gradient as functions of the integration step. It appears that the error in the directional gradient increases dramatically compared to the error in the objective function when increasing the integration step. However, below a certain threshold value for the integration step the derivatives clearly converge to stable results. A value equal to 0.05 m was chosen for Δs, which corresponded to an estimated absolute precision for the objective function of the order of 10^{-4} s.

In order to obtain comparable results from the optimisation performed evaluating derivatives by means of numerical differentiation, the same integration step was used. The optimisation algorithm chooses automatically either forward and centred difference formulae, the latter being usually applied in the vicinity of the solution, when accuracy becomes more important. Furthermore, the optimisation algorithm computes the value for the perturbation for the independent variables automatically on the basis of the expected accuracy of the objective function. Since a fixed step integrator is used, it is not possible to supply this information precisely and a few trials had to be done in order to achieve the best possible results for the established integration step.

7.4.3. Results

The same optimisation problem has been solved several times while steadily decreasing the optimality tolerance in order to find the maximum precision achievable using both differentiation methods. All the optimisation cases are solved starting from the same initial guess for the solution. The initial value for the objective function was 4.829 [s]. Table 7.2 reports the numerical results, i.e. the number of iterations, the CPU time (referring to a single processor workstation Alpha Digital EV6 at 700 MHz with 512 Mb of RAM, Unix operating system and the "gcc" compiler version 2.95.2) and the final objective function values, obtained for four values of optimality tolerance. The program using finite differences failed to converge for tolerance smaller than 10^{-3}. The optimisation algorithm stopped as it was unable to improve the objective function further to meet the required tolerance. Furthermore the second run stopped after only 52 iterations, returning a larger value for the objective function compared to the analogous case using AD, indicating that the corresponding tolerance is too large to obtain consistent optimisation results.

	AUTOMATIC DIFFERENTIATION			FINITE DIFFERENCES		
OptToll	*n iter.*	*CPU time*	*obj*	*n iter.*	*CPU time*	*obj*
2.0e-3	42	263 s	4.213 s	42	3137 s	4.213 s
1.0e-3	62	378 s	4.201 s	52	4206 s	4.207 s
5.0e-4	93	563 s	4.194 s	133 *	12616 s	4.194 s
2.0e-4	115	746 s	4.193 s	-	-	-

Table 7.2 Summary of the results (did not converge).*

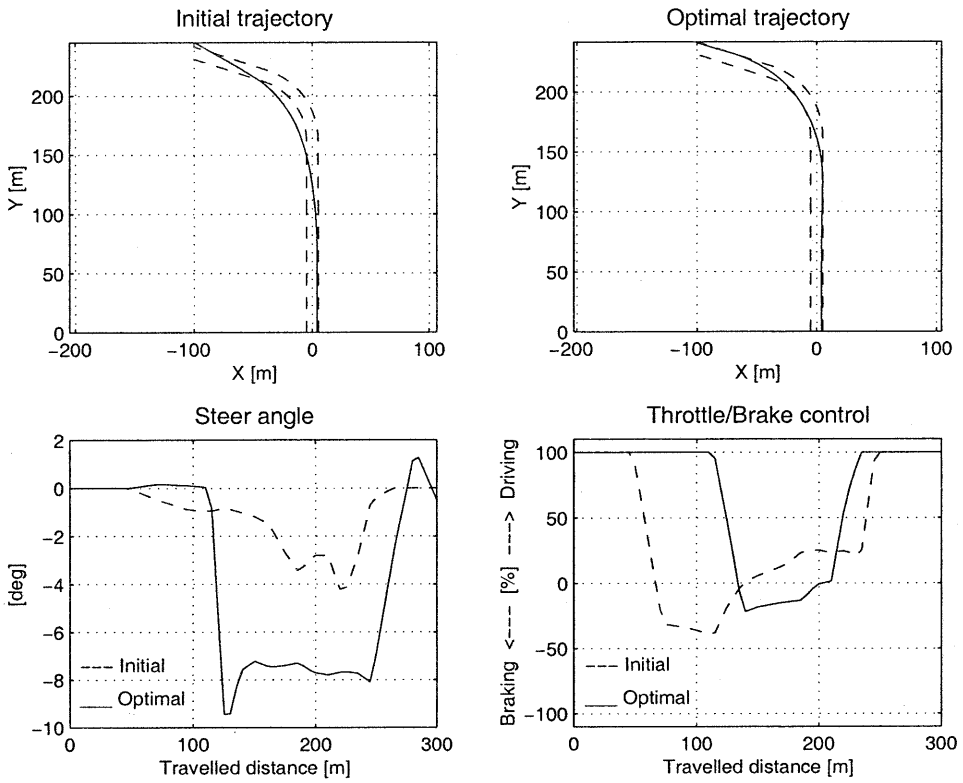


Figure 7.8 Optimisation results.

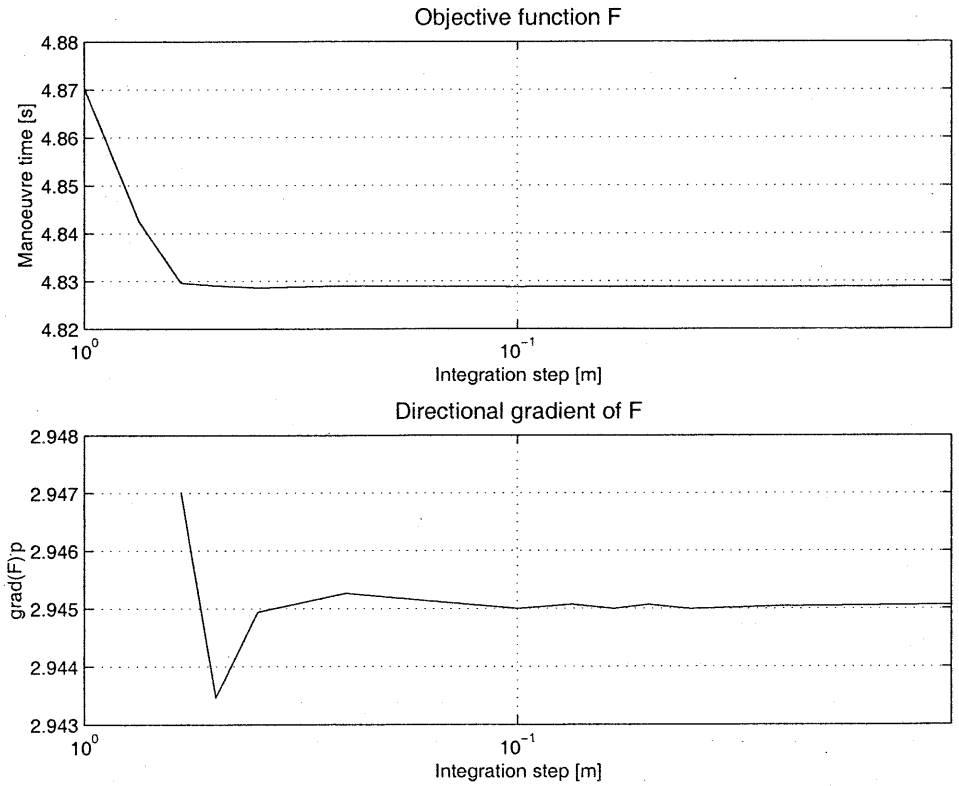


Figure 7.9 Convergence test for the identification of the integration step.

In summary, the optimisation program using AD is more than ten times faster compared to the one using numerical differentiation. Furthermore, it is more accurate thanks to the lack of truncation errors. Finally, in this simple example a convergence test for the optimal solution with respect to the density of the control discretisation grid was not included. This matter will be discussed in the next chapter. However, it must be said that one of the strengths of AD is that the computational time is nearly insensitive to the number of independent variables, giving the possibility to apply as many control decision points as needed without penalising the computational performance.

7.5. CONCLUSIONS

In this chapter the differentiation methods for obtaining derivatives, particularly for solving large scale optimisation problems, have been described. Particular emphasis has been put on Automatic Differentiation, and its application in the lap time optimisation program has been highlighted. By no means was this intended to be a rigorous treatment of AD theory. Instead, this was a report on the experience of approaching this methodology from scratch and applying it to a practical problem. The simple example presented here shows the enormous benefit which AD may provide to the solution of optimisation problems. The gain is certainly worth the slightly increased program complexity.

AD is not a new technique, it has been around for decades and it has been re-discovered and implemented several times. Modern programming languages make it now more attractive and easy to use for non-computer experts. The field is still a subject of intensive research, and the AD research community is now moving along with the optimisation research community to provide more efficient tools. Besides greater computational efficiency and accuracy, AD provides further possibilities, such as the automatic analysis of the problem structure, such as the degree of non-linearity or the exploitation of sparsity. Ultimately, with AD it is also possible to obtain higher order derivatives at a very cheap computational cost, which makes faster optimisation methods based on second order algorithms, such as Newton methods, more practical. It is foreseeable that the acronym "AD" will become more and more common among the people involved with optimisation.

Yet, there is already more technology available which we could have used, had it not been for the lack of time. The two main areas of interest for our optimisation problem are the handling of the look up tables, and the differentiation of the numerical integration algorithm. With regard to the first problem, there are object oriented AD tools, such as ADOL-C[2], where indexes of arrays can be active variables. In this way, when updating the independent variables in the computational graph, the indexes for the tables are updated as well. Also, the source-to-source transformation approach for AD offers the possibility to handle look-up tables. In principle, when applying this method, the newly generated code, augmented for AD, will embed also the operations which are necessary to compute the indexes for the look-up tables.

When the objective function contains the numerical solution of a set of ordinary differential equations, different strategies may be applied. The one presented here is the more straightforward, even though the least efficient. In summary, we used a fixed step integrator and we recorded the *entire* solution process on the computational graph. This

[2] Web site: `http://www.math.tu-dresden.de/wir/project/adolc/`.

approach works, but leads to very large data structures to handle. A more sophisticated strategy (Eberhard and Bischof, 1999) involves using AD to differentiate only the right-hand side of the ODE system, obtaining the sensitivities of the state derivatives with respect to the independent variables. Then, the sensitivities are integrated alongside the state derivatives and this allows to obtain state trajectories as well as their sensitivity with respect to the independent variables. Since the objective function is normally a function of the states, the chain rule applies. Furthermore, a variable step integrator may be used, in such a way to ensure not only the accuracy of the states, but also of the state derivatives.

More details about these possible developments will be given in the final conclusions. As far as the present work is concerned, all the elements are now in place for introducing the lap time optimisation program.

Chapter 8.

Introducing FastLap

In this chapter the implementation of the lap time optimisation algorithm in a computer program called FastLap is described. Firstly, the structure of FastLap will be outlined. Then, a brief description of the optimisation algorithm SNOPT® (Gill, Murray and Saunders, 1997), which is used here, will be given. The target will be to highlight the most important aspects of SNOPT which relate to the performance of the lap time optimisation algorithm. The current version of FastLap is built around a seven degrees of freedom vehicle model. This vehicle model was already outlined in §5.1, and will be extensively described in Appendix A. Here, its basic structure will be recalled, as it relates to some aspects of the implementation of FastLap. Finally, issues regarding the structure of the optimisation problem, such as the sparsity pattern of the Jacobian, will be addressed.

The development of a program such as FastLap may be viewed like the design and development of a racing car. Firstly, several months of design work are required to define the concept and the layout of the racing car. Then an equivalent amount of effort is required on the track, to fine tune the car and achieve the best of its performance. All the design work which led to the general layout of FastLap has been described in the previous chapter. Now the initial product is ready for its shakedown run and the subsequent phase of its development and fine tuning.

More than two years have passed since the first simple minimum time vehicle manoeuvring problem was successfully solved in the course of this project. Many trials and tests have been conducted to reach the present level of accuracy and robustness of FastLap, and it would be impossible to report them all. Hence, the aim here is to describe only the methodology which has been applied, hoping that this will be a useful guide for the further development of the lap time optimisation program.

8.1. Overview of FastLap

FastLap consists of two modules. The first module is a Matlab® Graphical User Interface (GUI), which is used for pre-processing the data needed for setting up a lap time optimisation problem, and for post-processing and visualising the results. The second module consists of two executable programs, which have to be run from the computer console: `fastlap`, the actual lap time optimisation program, and `vehicle7`, the simulation program for the seven degrees of freedom vehicle model.

8.1.1. The FastLap GUI

Running "`fastlap`" at the Matlab command prompt opens the FastLap GUI, which consists of two windows and a command palette. The main window, shown in figure 8.1, allows firstly for the selection of the track map. Upon loading a track data file, the circuit map is displayed together with some information, such as the circuit length, half the road width and the number of sectors in which the circuit is divided. A field for the selection of the vehicle model is also included, even though at the moment only the seven degrees of freedom model is available. Finally, a field for entering simulation parameters is included in the main window. Since a fixed step integrator is used, only one parameter is available, i.e. the print frequency. This parameter tells the simulation program `vehicle7` how often it is required to write the results on a file, e.g. print frequency equal to 20 implies to write the results every 20 integration steps. The second window, shown in figure 8.2, displays all the vehicle design and set-up parameters which are relevant for the selected vehicle model. In the bottom right hand corner, a slider is also provided, which allows to re-scale the tyre forces. This is necessary if one wants to compare the theoretical results with the performance measured on track, as the level of friction which is available on the circuit is usually different from that available in the laboratory conditions where the experimental tyre data are obtained.

In order to prepare the data file for one lap time optimisation case, the user must accomplish the following operations, using the controls on the command palette:

- **Load Circuit Map**; after selecting the circuit in the main window of the GUI, this command loads a file which embeds not only the map of the track, but also other information regarding the state and control discretisation scheme, the location of the road boundary constraints, the initial vehicle controls and the corresponding initial state trajectories. Furthermore, different values for the integration step of the vehicle dynamics equations are used for different trajectory segments, and this information is also embedded in the track file. We shall describe the track files more in detail later on.
- **Load SetUp**; this command allows to select and load a file with the values for the vehicle design and set up parameters. Upon loading these data, the values for the parameters are visualised in the second window of the GUI. The user can subsequently vary the parameters individually.
- **Load Tyres**; this command allows to select and load a file with the parameters needed by the tyre model used in the vehicle model.
- **Load Engine**; this command allows to select and load the experimental engine output map which is used in the mathematical model of the vehicle power train.
- **Load Aero Map**; this command allows to select and load the experimental aerodynamic map, which yields the values of the lift and drag coefficients as functions of the vehicle ride height. However, this command is not available with the seven degrees of freedom model, which uses constant drag and lift coefficients.
- **Make Data File**; when all the necessary data have been loaded and the vehicle set up parameters have been decided, this command allows to save the data file needed to run the `fastlap` executable program and begin the lap time optimisation. The data file is a formatted text file where all the data are written in a pre-determined order. At the start of an optimisation run, `fastlap` reads this data file sequentially and initialises all the variables which define the problem to be solved.

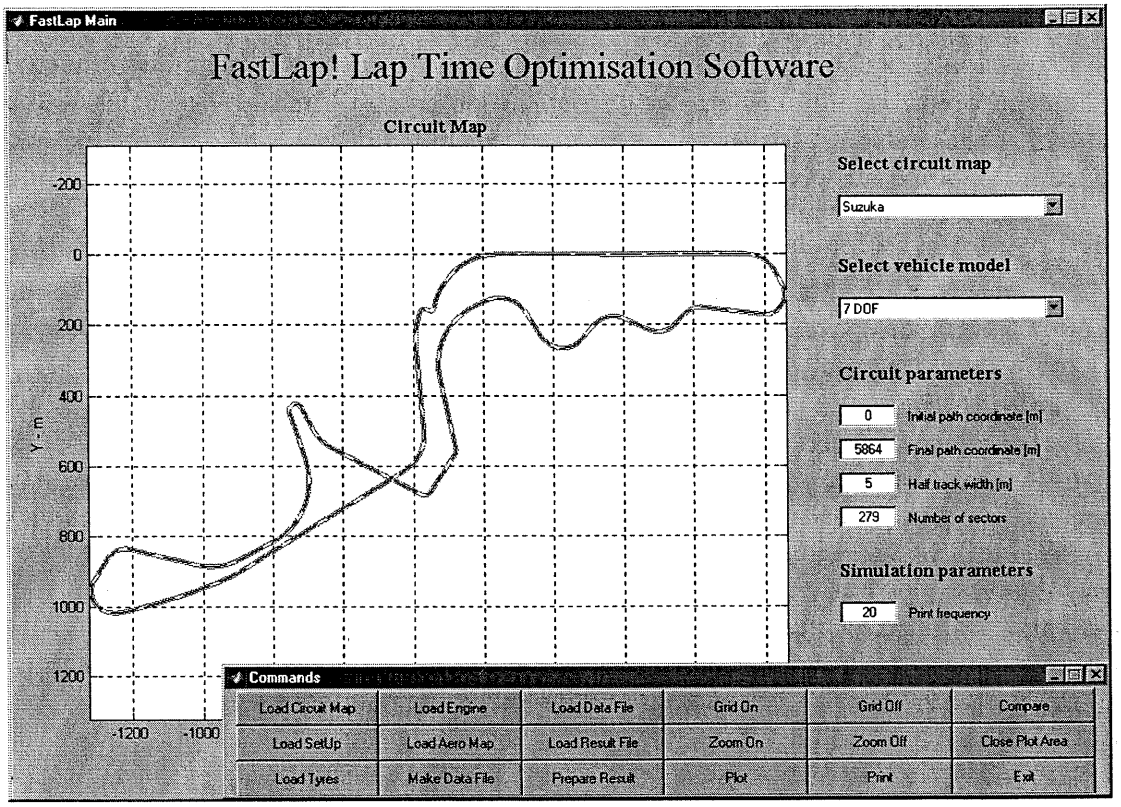


Figure 8.1 The FastLap GUI main window.

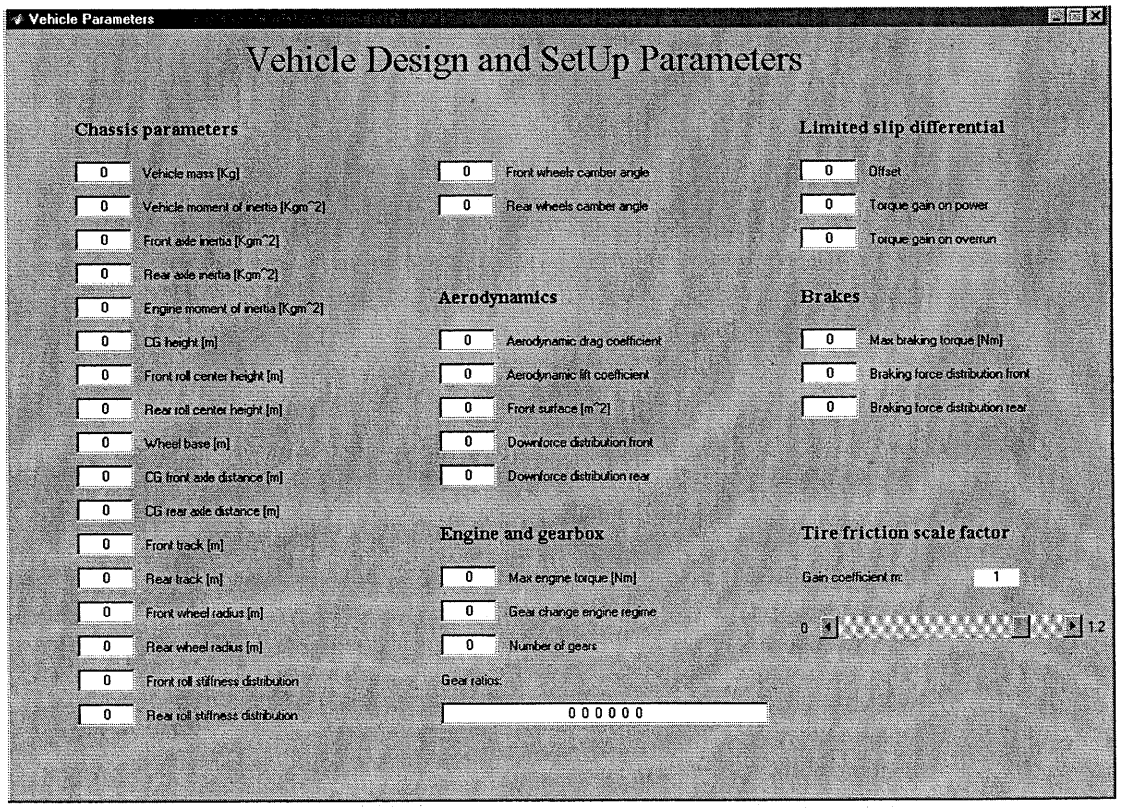


Figure 8.2 The FastLap GUI vehicle parameters window.

At the end of each optimisation run, `fastlap` saves the vector of optimal control and state parameters in a new text file. Next, the vehicle model may be simulated loading the optimal solution obtained, using the program `vehicle7` to generate the result file. The result file is yet another formatted text file which may be loaded and viewed in Matlab by using the FastLap GUI in order to analyse the optimal solution. In order to do so, the user must use the following commands:

- **Load Data File**; this command allows to select and load the data file for the optimisation case whose results are to be analysed. All the data which define the optimisation case are loaded back into the Matlab workspace.
- **Load Result File**; this command allows to select and load the set of results to view.
- **Prepare Results**; further calculations may be performed in order to analyse the results, and some of them may be time consuming, like the evaluation of the tyre saturation level. Hence, this command opens a selection window and the user may choose among six groups of graphs, depending on the need.

When the post-processing of the results is completed, two new windows appear; the FastLap plotter and the smaller plot selection window. In figure 8.3 these two windows are shown, with the plot selection window including all the six groups of graphs available.

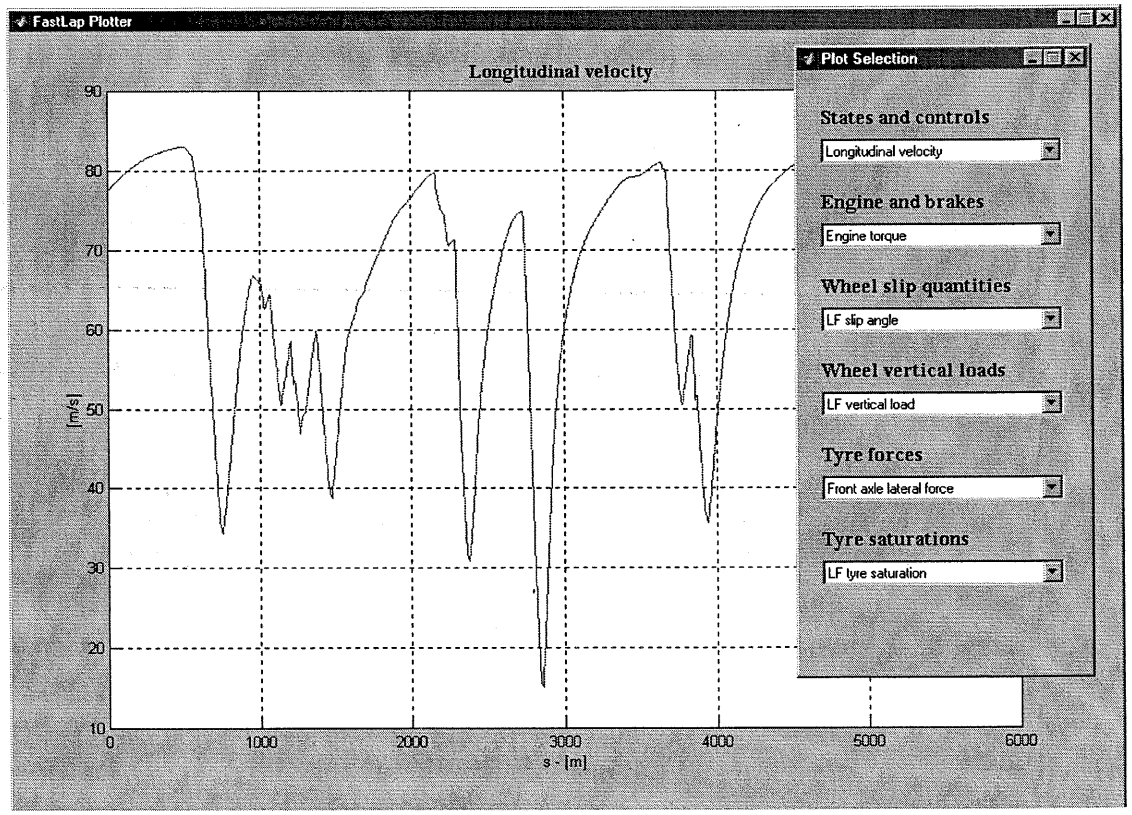


Figure 8.3 The FastLap plotter and plot selection window.

In order to view any of the available graphs, the user must firstly select the graph name from any of the six pop-up menus in the plot selection window. Then, pressing the command "plot" on the command palette causes FastLap to update the plotter with the new graph and it also brings the plotter in foreground. The user can also operate the usual Matlab plot options "grid" and "zoom" from the command palette. The command "print" allows to save on disk an Encapsulated PostScript file of the graph. The command "compare" is used to load a second result file, to which the same post-processing as the first file is automatically applied. Using the "plot" command subsequently allows the direct comparison of the two result files on each graph. Finally, the command "exit" closes all the windows.

8.1.2. The FastLap executable programs

The lap time optimisation algorithm is implemented in the `fastlap` executable program. The main body of this program is written in the C/C++ programming language. It consists of three major modules, which are:

- The vehicle dynamics equations, which are written in a C/C++ function and are generated using the symbolic multibody code AutoSim®, see Appendix B.
- The Automatic Differentiation library **AD***opt*, which was described in Chapter 7.
- The optimisation algorithm SNOPT, which consists of a set of FORTRAN 77 subroutines and is the core of FastLap.

To run `fastlap`, the user must type the following command line at the computer console:

```
fastlap <data file> <optim. file> <options>
```

Here, `<data file>` is the name of the file generated using the GUI which contains all the necessary data to define an optimisation case, and `<optim. file>` is the name of the file where `fastlap` will save the optimal control and state parameters at the end of the run. Different strategies for the lap time optimisation may be selected by passing certain options as input arguments to `fastlap`. These options will be described in detail later on, as it will firstly be necessary to describe the implementation of the lap time optimisation program more in details.

The program `vehicle7` is a self-contained simulation program which has been generated using AutoSim. The program has been modified in such a way that it may use the same data files as `fastlap`. There are two options for calling `vehicle7`:

```
vehicle7 <data file>
```

and

```
vehicle7 <data file> <optim. file>
```

In the first case, the vehicle model will be simulated using the initial control and state parameters which are included in the data file. Instead, in the second case the vehicle will be simulated using an optimal solution obtained from `fastlap`.

8.2. OVERVIEW OF SNOPT

A brief overview of the optimisation algorithm SNOPT is given here. For more information, extensive documentation of this software can be found at the SNOPT web site: `www.sbsi-sol-optimize.com/SNOPT.htm`.

SNOPT is a general purpose system for solving optimisation problems involving many variables and constraints. It minimises linear or non-linear functions subject to bounds on the variables and general linear or non-linear constraints. It is particularly suitable for large-scale problem with sparse constraints, as it is designed to take advantage of the sparse structure of the Jacobian.

SNOPT finds solutions that are *locally optimal*, and ideally any non-linear function should be smooth. However, local discontinuities can be tolerated, as long as they are not too close to the solution. For general non-linear constrained optimisation problems it is generally better if the user can provide the problem derivatives. SNOPT allows the freedom to specify all or part of the derivatives, and it will automatically estimate the remaining ones by means of finite differences.

SNOPT uses a Sequential Quadratic Programming (SQP) algorithm that obtains search directions from a sequence of Quadratic Programming (QP) sub-problems. Each QP sub-problem minimises a quadratic model of a certain Lagrangian function subject to a linearisation of the constraints, see §2.4.2. An augmented Lagrangian merit function (Gill, Murray and Saunders, 1997) is reduced along each search direction to ensure convergence from any starting point. Finally, SNOPT allows *infeasible* intermediate solutions. That is, intermediate solutions which violate the constraints.

SNOPT is most efficient if only some of the variables enter non-linearly, or if the number of active constraints, including simple bounds, is nearly as large as the number of variables. SNOPT usually requires relatively few evaluations of the problem functions. Hence it is especially effective if the objective or constraint functions (and their derivatives) are expensive to compute.

The source code for SNOPT is written in FORTRAN 77. SNOPT may be called from a driver program (typically in FORTRAN as well, or in C/C++, such as FastLap, or even in Matlab). Before an optimisation run may start, the user has the possibility to change the values of many setting parameters for SNOPT. A brief review of some of these parameters which have been more relevant during the development of FastLap is given below.

- **Major iteration limit**; the basic structure of SNOPT involves *major* and *minor* iterations. The major iterations are those which generate the sequence of intermediate solutions which converge to a point that satisfies the first-order conditions for optimality, see Eq. 2.56. Each major iteration involves the solution of a QP sub-problem to generate the search direction. This is also an iterative procedure which involves the so-called minor iterations. The default value for the major iteration limit is equal to $\max\{1000, n_c\}$, where n_c is the total number of problem constraints. In FastLap, though, the limit has been set equal to only 200. This is because the objective and constraint functions (and their derivatives) in FastLap are very expensive to compute, and the chosen value for the major iterations limit is intended to bound the maximum total time for the solution of one optimisation problem within practical values. On the other hand, this sets the challenge for setting up a well-behaving optimisation problem which can be solved

with sufficient accuracy within the maximum number of iteration allowed.

- **Derivative or non-derivative line search**; at each major iteration a line search is performed to find the step length τ along the search direction which produces a sufficient decrease in the merit function, see §2.4.2. SNOPT offers two options for the line search algorithm. A derivative line search uses a cubic interpolation and requires both function and derivatives in order to compute estimates of the step τ. A non-derivative line search, instead, uses a quadratic interpolation and only requires function values. While the former method should be more accurate, hence reducing the number of major iterations required to reach the optimal solution, the time overhead due to the extra evaluation of the problem derivatives was overwhelming compared to the benefit. Hence, in FastLap the non-derivative line search option is set as default.
- **Line search tolerance**; SNOPT allows to control the accuracy with which the step length is located along the search direction. At the start of each line search a target directional derivative for the merit function is computed. In principle, the minimum of the merit function along the search direction lies where its directional derivative becomes perpendicular to the direction of search. The line search tolerance determines the accuracy with which such target directional derivative is approximated. Its value must be in the range [0.0, 1.0]. An accurate line search (tolerance equal to 0.1 or less) is convenient when the problem functions are cheap to evaluate. Conversely, a less accurate line search (tolerance equal to 0.9 or 0.99) is advisable when the problem functions are computationally expensive. The lower accuracy may require more major iterations, but the total number of function evaluations may decrease to compensate. Varying this parameter in FastLap has a tangible effect on the solution time of a lap time optimisation problem. In the end, a moderately accurate line search (tolerance equal to 0.9) gave the best results, not only in terms of computational time, but also in terms of robustness. This is probably due to the fact that if the problem functions (and their derivatives) cannot be computed with a high level of accuracy, which is the case in FastLap where the numerical integration of the vehicle dynamics equations is involved, high accuracy can not be achieved in the line search.
- **Major step limit**; this parameter allows to set a bound for the step length τ in the line search. This is very useful in problems where the problem functions may, in certain conditions, become very sensitive with respect to tiny variations of the independent variables. Normally, the step length τ is included in the range [0, 1]. However, an extra upper bound is defined as follows:

$$\beta = s_{\text{lim}}\left(1 + \|\mathbf{x}\|\right)/\|\mathbf{p}\| \qquad \text{Eq. 8.1}$$

Here, $\mathbf{x}$ is the vector of independent variables, $\mathbf{p}$ is the search direction and s_{lim} is the major step limit. The upper boundary for the step length is then chosen to be equal to $\min\{1,\beta\}$. If the sensitivity of the problem functions becomes very large, the norm of the search direction will be very large as well. Consequently β may become much smaller than 1, and the user may interfere by adjusting the value of the major step limit. The default value for this parameter is 2.0, which makes β hardly ever the limiting factor for the step length. However, for special problems, setting the step limit equal to 0.1 or even less can significantly improve the robustness of the

optimisation, even though it slows down the convergence. This relates very well with the lap time optimisation problem, where the vehicle states may become very sensitive to small variations in the controls when the tyres are operated on the edge of their friction limit, particular when a vehicle is biased towards oversteer.

- **Major feasibility tolerance**; this specifies how accurately the non-linear constraints have to be satisfied at the solution, and is the first of the two conditions which have to be met for the successful completion of an optimisation run.
- **Major optimality tolerance**; this is the second condition which has to be met for the successful termination of an optimisation run and relates to the accuracy of the optimal solution. The measure of such accuracy relates to the necessary conditions for optimality and is based on the so-called *reduced gradients*. In simple words, when all the reduced gradients are equal to zero (or sufficiently close to zero) it means that the components of the gradient of the objective function are either equal to zero or there are active constraints which prevent any further decrease of the objective function unless any of such constraints are violated. For a formal definition of the feasibility and optimality conditions, the reader is referred to the documentation of SNOPT.

As far as the development of FastLap is concerned, the task will be to achieve the greatest accuracy for the optimal solution which is compatible with the accuracy which is yielded by the simulation of the vehicle dynamics equations and the Automatic Differentiation of the integration algorithm, while also keeping the computational time within acceptable values.

8.3. IMPLEMENTATION OF FASTLAP

The general lap time optimisation algorithm was described in chapter 6. Here, its implementation in FastLap using the seven degrees of freedom vehicle model is described. First of all, let us recall the general definition of the lap time optimisation algorithm. The task is to find the vector of the independent optimisation parameters $\mathbf{y}$ which solves the following Non-Linear Programming problem:

$$\min_{\mathbf{y}} \quad J(\mathbf{y})$$

$$\begin{aligned} &\textit{subject to}: \\ &\textit{Continuity constraints}: \quad \xi_i(\mathbf{y}) = 0 \qquad i = 1,\ldots,m \\ &\textit{Road constraints}: \quad c_i(\mathbf{y}) \le 0 \qquad i = 1,\ldots,m_c \\ &\textit{Bounds}: \quad \mathbf{y}_L \le \mathbf{y} \le \mathbf{y}_U \end{aligned} \qquad \text{Eq. 8.2}$$

The vector $\mathbf{y}$ is defined as the combination of the independent vehicle control and state parameters:

$$\mathbf{y} = \mathbf{u}_n \cup \mathbf{x}_m \qquad \text{Eq. 8.3}$$

In Eq. 8.2, J is the objective function, i.e. the lap time, ξ_i is the vector of defects, i.e. the

measures of the discontinuities in the vehicle state trajectories at the junction of each track segment, and c_i is the measure of the constraint violation at each road boundary check point. Finally, suitable bounds apply to all the optimisation parameters.

8.3.1. The vehicle model layout in relation to FastLap

The vehicle model in FastLap has seven degrees of freedom and two control inputs. The chassis is treated as a rigid body with three degrees of freedom, the yaw angle and the lateral and longitudinal displacements. The wheels are rigidly attached to the body, allowing only the rotation of each wheel relative to the vehicle chassis about its spin axis. Therefore, four more degrees of freedom are added to the system.

The two control inputs are the vehicle lateral control, i.e. the front road wheel steer angle, and the vehicle longitudinal control. A single variable is defined for the longitudinal control, which may assume values in the range [-1, 1]. It is assumed that this variable represents the throttle aperture when it is positive, or a portion of the maximum braking torque available when it is negative.

The mathematical model of the vehicle consists of fourteen first-order differential equations. According to the parallel shooting algorithm, the vehicle trajectory is divided in m totally de-coupled segments. Within each segment the vehicle trajectory is evaluated by solving the following initial value problem:

$$\frac{d\mathbf{x}}{ds} = \bar{\mathbf{a}}(\mathbf{x}(s), \mathbf{u}(s), s)$$

$$with: \quad \mathbf{x}(s_{i-1}) = \mathbf{x}_{i-1} \qquad s \in [s_{i-1}, s_i] \qquad \text{Eq. 8.4}$$

$$for: \quad i = 1, \ldots, m$$

The vector $\mathbf{x}$ of the system state variables includes the following components:

- $\mathbf{x}(1)$ yaw angle;
- $\mathbf{x}(2)$ yaw rate;
- $\mathbf{x}(3)$ longitudinal velocity;
- $\mathbf{x}(4)$ lateral velocity;
- $\mathbf{x}(5)$ x co-ordinate of the centre of gravity;
- $\mathbf{x}(6)$ y co-ordinate of the centre of gravity;
- $\mathbf{x}(7)$ front left wheel angular position;
- $\mathbf{x}(8)$ front left wheel angular velocity;
- $\mathbf{x}(9)$ front right wheel angular position;
- $\mathbf{x}(10)$ front right wheel angular velocity;
- $\mathbf{x}(11)$ rear left wheel angular position;
- $\mathbf{x}(12)$ rear left wheel angular velocity;
- $\mathbf{x}(13)$ rear right wheel angular position;
- $\mathbf{x}(14)$ rear right wheel angular velocity;

In the general case it was assumed that each set of independent vehicle state parameters

at each trajectory junction includes all the vehicle states. In reality this is unnecessary. Firstly, the continuity of the wheel angular positions is not required across a junction. Secondly, instead of specifying the vehicle x and y position at the junction, we may simply specify the distance of the vehicle centre of mass from the point on the road centre line identified by the co-ordinates x_t, y_t at s equal to s_{i-1},, as shown in figure 8.4.

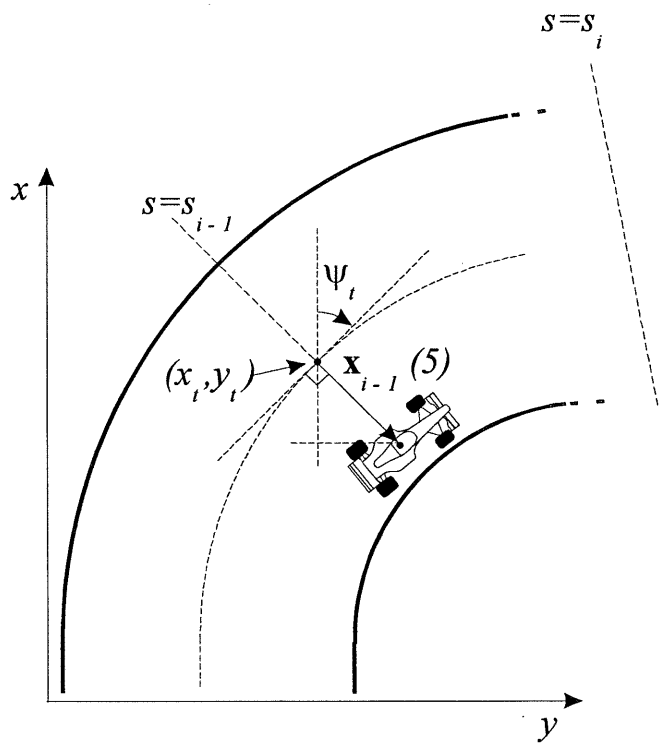


Figure 8.4 Evaluation of the vehicle initial position.

Hence, only nine state parameters are necessary and sufficient to define the initial conditions for the vehicle at the beginning of each segment. These parameters are:

- $\mathbf{x}_{i-1}(1)$ initial yaw angle;
- $\mathbf{x}_{i-1}(2)$ initial yaw rate;
- $\mathbf{x}_{i-1}(3)$ initial longitudinal velocity;
- $\mathbf{x}_{i-1}(4)$ initial lateral velocity;
- $\mathbf{x}_{i-1}(5)$ initial distance of the vehicle centre of mass from road centre line;
- $\mathbf{x}_{i-1}(6)$ initial front left wheel angular velocity;
- $\mathbf{x}_{i-1}(7)$ initial front right wheel angular velocity;
- $\mathbf{x}_{i-1}(8)$ initial rear left wheel angular velocity;
- $\mathbf{x}_{i-1}(9)$ initial rear right wheel angular velocity;

The initial conditions for each segment may then be expressed as functions of the nine state parameters defined above, and Eq. 8.4 changes as follows:

$$\frac{d\mathbf{x}}{ds} = \bar{\mathbf{a}}(\mathbf{x}(s), \mathbf{u}(s), s)$$

$$with: \quad \mathbf{x}(s_{i-1}) = \mathbf{f}(\mathbf{x}_{i-1}) \qquad s \in [s_{i-1}, s_i] \qquad \text{Eq. 8.5}$$

$$for: \quad i = 1, \ldots, m$$

Here, the vector function $\mathbf{f}(\mathbf{x}_{i-1})$ at $s = s_{i-1}$, reads:

$$\begin{cases} \mathbf{x}(1) = \mathbf{x}_{i-1}(1) \\ \mathbf{x}(2) = \mathbf{x}_{i-1}(2) \\ \mathbf{x}(3) = \mathbf{x}_{i-1}(3) \\ \mathbf{x}(4) = \mathbf{x}_{i-1}(4) \\ \mathbf{x}(5) = x_t(s_{i-1}) - \mathbf{x}_{i-1}(5) \cdot \sin(\psi_t(s_{i-1})) \\ \mathbf{x}(6) = y_t(s_{i-1}) + \mathbf{x}_{i-1}(5) \cdot \cos(\psi_t(s_{i-1})) \\ \mathbf{x}(7) = \mathbf{x}(9) = \mathbf{x}(11) = \mathbf{x}(13) = 0 \\ \mathbf{x}(8) = \mathbf{x}_{i-1}(6) \\ \mathbf{x}(10) = \mathbf{x}_{i-1}(7) \\ \mathbf{x}(12) = \mathbf{x}_{i-1}(8) \\ \mathbf{x}(14) = \mathbf{x}_{i-1}(9) \end{cases} \qquad \text{Eq. 8.6}$$

The vector of defects ξ_i for a generic trajectory junction, which was defined in Eq. 6.28 for the general case, has as many components as the number of independent state parameters at that junction, and is defined as follows:

$$\begin{cases} \xi_i(1) = \mathbf{x}(1) - \mathbf{x}_i(1) \\ \xi_i(2) = \mathbf{x}(2) - \mathbf{x}_i(2) \\ \xi_i(3) = \mathbf{x}(3) - \mathbf{x}_i(3) \\ \xi_i(4) = \mathbf{x}(4) - \mathbf{x}_i(4) \\ \xi_i(5) = d_i - \mathbf{x}_i(5) \\ \xi_i(6) = \mathbf{x}(8) - \mathbf{x}_i(6) \\ \xi_i(7) = \mathbf{x}(10) - \mathbf{x}_i(7) \\ \xi_i(8) = \mathbf{x}(12) - \mathbf{x}_i(8) \\ \xi_i(9) = \mathbf{x}(14) - \mathbf{x}_i(9) \end{cases} \qquad \text{Eq. 8.7}$$

where $\mathbf{x} = \mathbf{x}(s_i)$ and d_i is the actual distance of the vehicle from the road centre line at the i^{th} junction.

8.3.2. Scaling

The term "scaling" is often used in a vague sense in the context of optimisation, to discuss numerical difficulties whose existence is universally acknowledged, but which cannot be described precisely. Scaling is used to develop FastLap with the essential task of obtaining a "well balanced" optimisation problem. Since the set of independent optimisation parameters includes variables of different nature, i.e. steer angle control input, throttle and brake control input and state variables, it is natural to expect that each of these groups of variables has a different weight on the objective and constraint functions. A well balanced optimisation problem is a problem where the sensitivities of the objective and constraint functions with respect to all the independent variables are similar in magnitude to each other.

The consequence of bad scaling is slow convergence. Intuitively, one may imagine that if the problem derivatives are badly scaled, at each iteration the search direction will be biased towards such derivatives as are greater in magnitude. Hence, while some variables will reach their optimal values rather soon, others will progress towards the solution at a much slower rate. Furthermore, in extreme cases of bad scaling where the problem derivatives differ by several orders of magnitude, numerical problems may arise, the Hessian matrix may become ill-conditioned and the optimisation may fail to converge at all. This does not normally happen in the lap time optimisation problem, except in the case where the vehicle reaches the tyre friction limit at the rear tyres first, as was extensively discussed in §6.3. However, the solution to this problem is not scaling, but the appropriate problem discretisation.

Scaling in FastLap is achieved by means of a linear transformation of the problem variables. Given the original vector of independent optimisation parameters $\mathbf{y}$, the transformed variables read:

$$\bar{\mathbf{y}} = \mathbf{D} \cdot \mathbf{y} \qquad \text{Eq. 8.8}$$

Here, $\mathbf{D}$ is a diagonal constant matrix. At the start of an optimisation run, the independent optimisation variables are initialised and scaled. Hence, the optimisation algorithm always sees the scaled variables and uses them to call the objective and constraint functions. Then, inside these functions, the variables are re-scaled to their original values, which are physically meaningful for the mathematical model of the vehicle. The effect on the derivatives may be summarised as follows. Consider, for simplicity, only the objective function and its gradient:

$$J = obj(\mathbf{y}) \qquad \text{Eq. 8.9}$$

Applying the transformation we obtain:

$$J = obj(\bar{\mathbf{y}}) = obj(\mathbf{D} \cdot \mathbf{y}) \qquad \text{Eq. 8.10}$$

Then, the derivatives of the transformed objective function relate to those of the original one as follows:

$$\frac{\partial J}{\partial \bar{\mathbf{y}}} = \frac{\partial J}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \bar{\mathbf{y}}} = \mathbf{D}^{-1} \cdot \frac{\partial J}{\partial \mathbf{y}} \qquad \text{Eq. 8.11}$$

The elements of the diagonal matrix **D** have been chosen in such a way to make the components of the vector on the left hand side of Eq. 8.11 as close as possible in magnitude with respect to each other. For a given trial vector **y** of optimisation variables, the gradient of the objective function J with respect to the un-scaled variables **y**, is computed. Then, the average value of its components is evaluated and is taken as target value for the scaled gradient of J, i.e. each element of the left hand side vector of Eq. 8.11 is set equal to the average value of the un-scaled vector gradient. Eq. 8.11 may finally be solved for **D**. Due to the non-linear behaviour of the vehicle, the ideal scaling may be different depending on how close the vehicle is to its performance limits. Therefore this procedure was repeated using trial vectors of optimisation variables which correspond to the vehicle being more or less close to its maximum performance, and a compromise was achieved. Finally, also the problem constraints may be scaled using the same technique described above. The task is again to balance the constraints with respect to each other in such a way that they have similar weights in the solution process. The choice of the scaling coefficients was done during the development of FastLap and their values are defined in the program as constants.

8.3.3. Sparsity pattern: the map of the Jacobian

The chosen solution method for the minimum lap time optimisation problem implies that the Jacobian of the constraints is sparse. SNOPT requires the sparsity pattern of the Jacobian in order to be able to compute each iteration step at much lower cost. For a given lap time optimisation problem, once that the control and state discretisation scheme is defined, alongside with the location of the road boundary check points, the map of the non-zero terms of the Jacobian can be derived. The procedure has been automated in the initialisation routine for FastLap. Here, with the aid of a simple example, we shall introduce the typical problem discretisation and describe such procedure.

First of all, it is necessary to define the partition of the dependent and independent variables in FastLap. This is almost totally arbitrary, and the following choice was made. The vector **y** of the independent optimisation parameter will be organised as follows:

- all the state parameters first;
- all the steer control input parameters second;
- all the throttle and brake control input parameters last:

$$\mathbf{y} = \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m-1}; \mathbf{u}_{stn}; \mathbf{u}_{tbn}\} \qquad \text{Eq. 8.12}$$

Then, the vector **c** of the problem constraints is so organised:

- all the road boundary constraints first;
- the vector of defects last:

$$\mathbf{c} = \{c_1, c_2, c_3, \ldots, c_{m_c}; \xi_1, \xi_2, \xi_3, \ldots, \xi_m\} \qquad \text{Eq. 8.13}$$

Consider the example shown in figure 8.5. The vehicle trajectory is divided in three

segments. For simplicity, we shall consider that the track is open, i.e. the vehicle final states do not have to coincide with the initial states. Furthermore, we shall consider that the vehicle initial states for the first segment are fixed. Therefore, there are two sets of vehicle state parameters, $\mathbf{x}_1$ and $\mathbf{x}_2$, and two sets of defects, ξ_1 and ξ_2 at the junctions of the three segments. It is assumed that the lateral and longitudinal vehicle controls share the same discretisation grid, which consists of ten control decision points. Furthermore, it is assumed that each segment junction coincides with a control decision point. In other words, the instances for the state discretisation grid are multiple in size of those of the control discretisation grid. Finally, only one road boundary check point is defined within each trajectory segment, as indicated in the figure. For simplicity, it is assumed that the segment junctions located at the apexes of the two turns are sufficient to enforce the road boundary constraints at those crucial locations, since the state parameters at the beginning of each segment are bounded in such a way that the vehicle trajectory may not lie outside. In practice, though, this is not acceptable in terms of accuracy, as we shall see a little later. Hence, according to the definitions given in §8.3.1, the problem shown in figure 8.5 has 38 independent optimisation parameters (2 times 9 state parameters plus 10 times 2 control parameters) and 21 constraints (2 times 9 defects plus 3 road boundary check points).

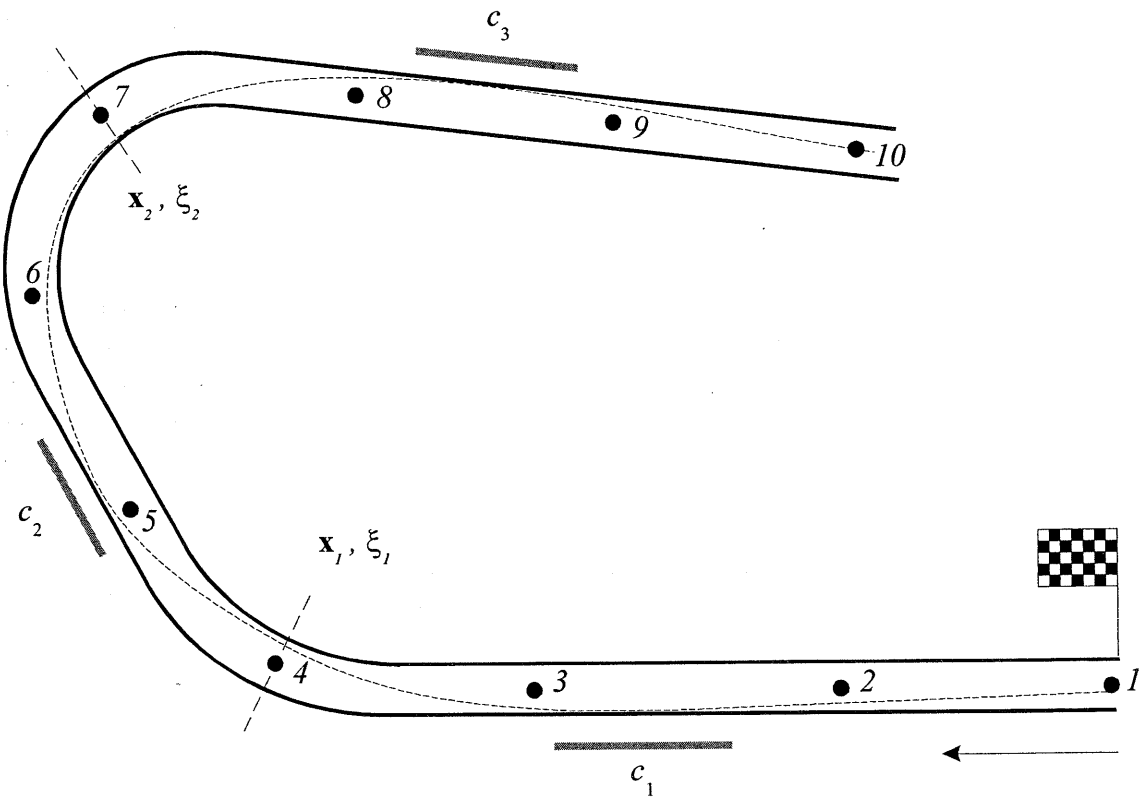


Figure 8.5 Problem discretisation example.

Figure 8.6 shows the map of the Jacobian, with the grey cells representing the non-zero elements. A clear pattern, which is repeated for the groups of independent and dependent variables related to the various trajectory segments, is visible. Let us now examine in detail the mutual dependencies between the various groups of variables.

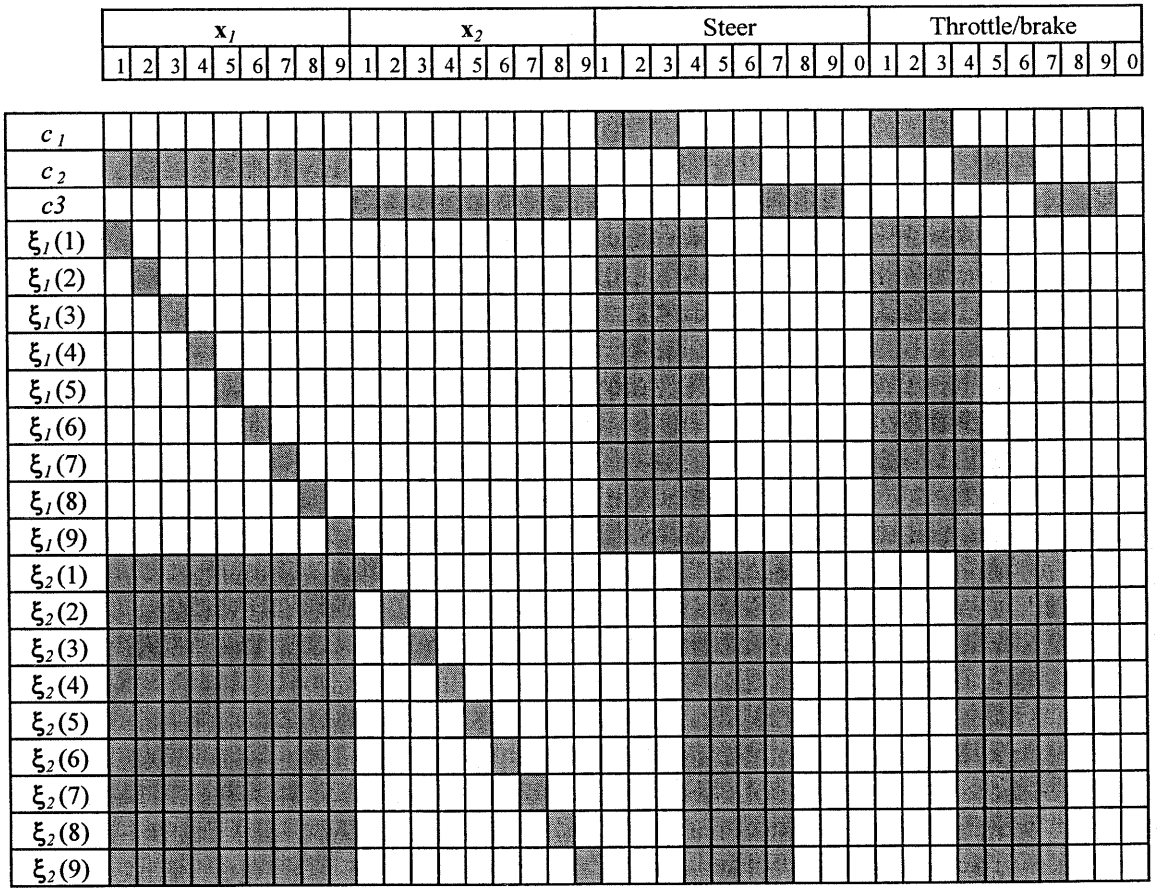


Figure 8.6 The map of the Jacobian.

Firstly, let us consider the state parameters, particularly the set of state parameters for the first junction $\mathbf{x}_1$. Each of these parameters affects the entire vehicle trajectory across the second segment. Hence, the road boundary constraint c_2 will change if any of the components of $\mathbf{x}_1$ is varied. Then, each of the components of the set of state parameters $\mathbf{x}_1$ is related to one component of the set of defects ξ_1, as is clear from Eq. 8.7. This determines the diagonal non-zero elements which is visible in the left hand side of the map of the Jacobian, see figure 8.6. Finally, each of these state parameters will affect all the defects ξ_2, evaluated at the end of the second segment, and this determines the dense part at the bottom left hand corner of the map of the Jacobian.

Let us now consider the vehicle control inputs. In order to evaluate their effects on the problem constraints, it is necessary to refer to the interpolation method which is used to evaluate the actual control input to the vehicle model. In this case we shall consider that a linear interpolation technique is used. Hence, when varying the i^{th} control decision point, the actual variation of the input control history (and of the vehicle state trajectories) extends from the previous control decision point, the i^{th}-1, to the next point, the i^{th}+1, with obvious exceptions for the first and last points. Therefore, the control decision points number 1, 2 and 3 affect the first road boundary constraint, the points number 4, 5 and 6 the second and so forth. Finally, with regard to the defects, the points number 1, 2, 3 and 4 will affect all the vehicle states at the end of the first segment, and consequently all the defects. The control point number 4, which occurs at the junction, will also affect the next segment, together with the points number 5, 6, and 7. This explains why the largest blocks of non-zero elements in the right hand side of

the map of the Jacobian overlap in correspondence of the 4^{th} control variable, both for the steer angle and the throttle/brake control.

8.3.4. Lap time optimisation program flow charts

In this section the general layout of the lap time optimisation program is summarised, and the various options which have been built into the program are described by means of two flow charts. Figure 8.7 shows the flow chart for the entire lap time optimisation algorithm. In FastLap all the constant parameters, such as the vehicle model parameters and the optimisation algorithm set-up parameters, are held in global data structures. This is convenient in order to make all the constant problem parameters available to all the program routines, without having to write long lists of input arguments for calling them. At the start, FastLap performs all the necessary initialisation. This includes reading the data file, copying the values of the parameters into the global data structure and performing further analysis, such as deriving the map of the Jacobian from the data which describe the discretisation scheme. Next, the initial solution vector $\mathbf{y_0}$ is assembled, according to the partition defined by Eq. 8.12, and the pre-defined linear transformation for scaling the independent variables is applied. At this point FastLap offers two options. One is to start the optimisation with the nominal initial guess for the solution, which was included in the data file, and the other is to vary the initial guess by applying random noise of suitable magnitude to the vector of independent optimisation variables. This feature has been extensively used to test the repeatability of an optimal solution and to investigate the presence of local minima, as we shall see in the next chapter. The actual optimisation may then begin by calling the optimisation algorithm. SNOPT takes the initial solution vector and starts by calling the objective function, where the lap simulation takes place, for the first time to obtain the values of the lap time and the constraints, and their derivatives.

Figure 8.8 shows the flow chart of the objective function. When the function is called, the current solution vector is installed and re-scaled, so that the optimisation variables re-gain their physical significance. Next, the lap simulation is performed using Automatic Differentiation arithmetic, as was described in §7.3, using the AD software library **AD***opt*. This involves firstly the recording of all the algebraic operations involved in the lap simulation on the computational graph, i.e. the `CodeList`, as is defined in **AD***opt*, and then extracting the required function values and/or derivatives using the appropriate routines defined in the AD software library. However, due to the limitations in **AD***opt*, the ideal situation where the computational graph is constructed only once, at the beginning of the optimisation, and re-used for all the necessary iterations by only updating the independent variables, could not be implemented. Instead, the trajectory segments which make up the entire lap are simulated sequentially. For each segment a new `CodeList` object is constructed. The independent optimisation variables which relate to the current segment are associated to the `CodeList` as active variables. Next, the simulation is performed using a fixed-step, second order Runge-Kutta integration formula, and the results for the segment are extracted from the `CodeList` and stored in temporary variables. Then, the current `CodeList` object is cleared and a new one is initialised for the successive segment, until all the m segments have been simulated. Only then are the temporary results assembled and output to SNOPT.

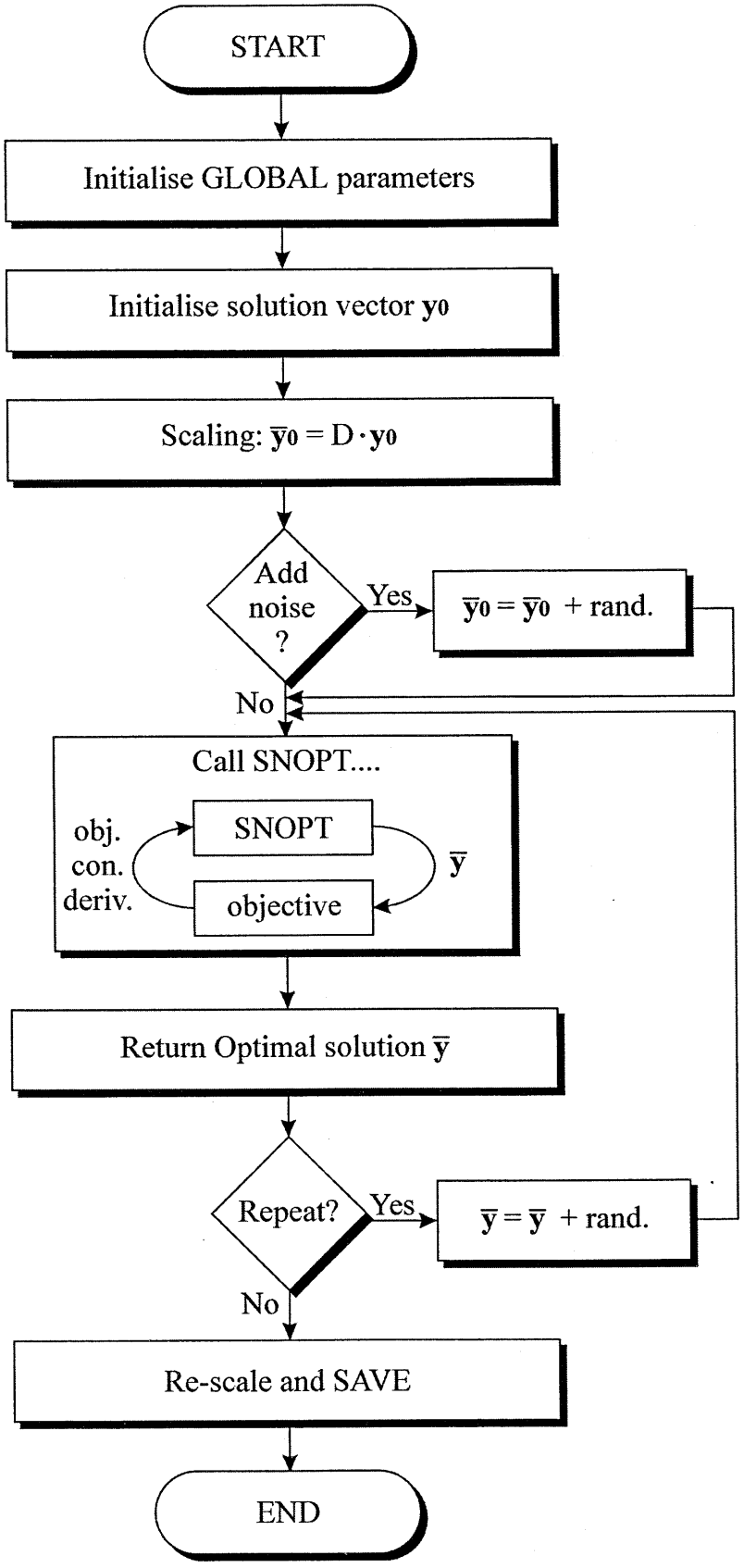


Figure 8.7 Lap time optimisation program flow chart.

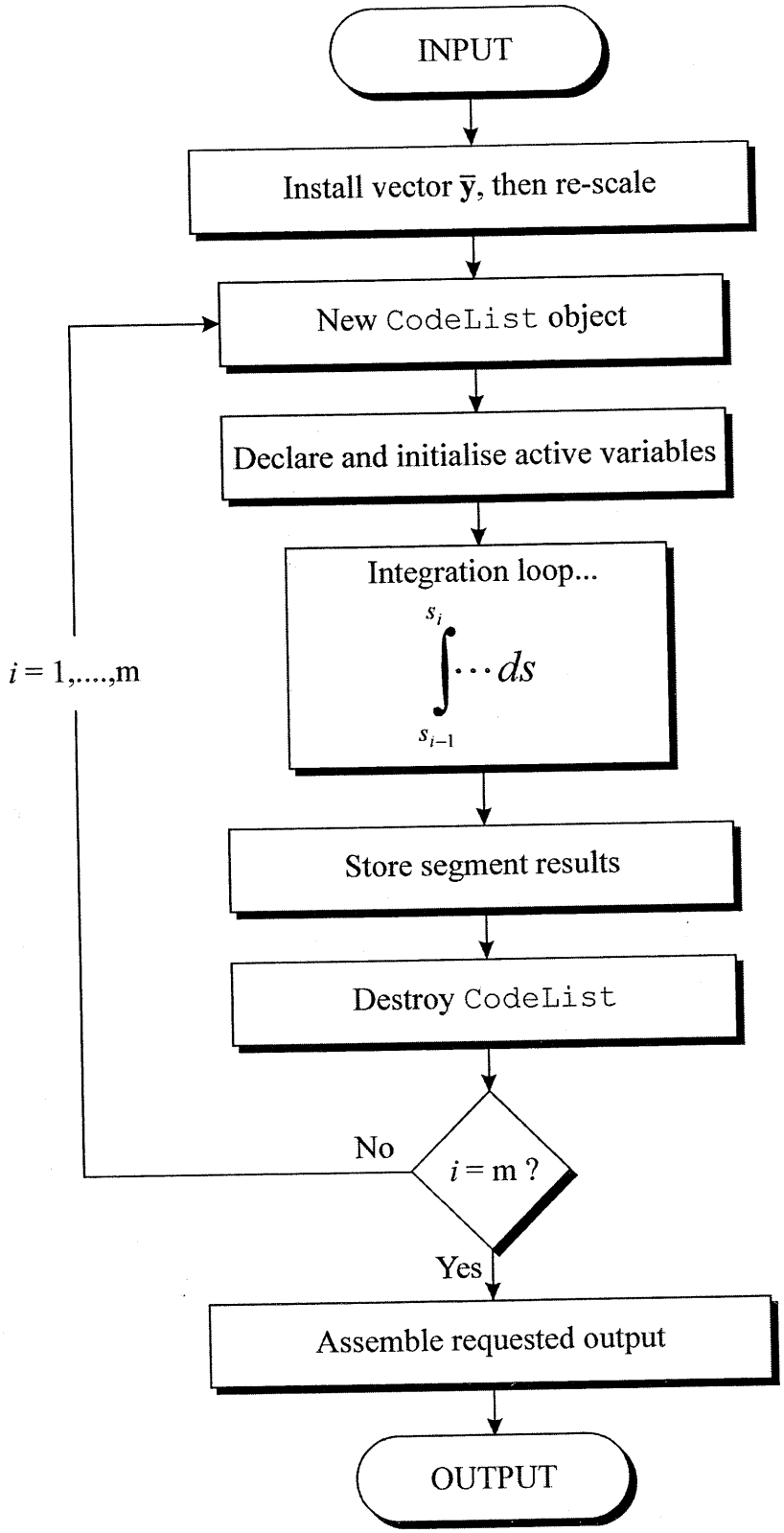


Figure 8.8 Objective function flow chart.

When the optimisation is completed, SNOPT returns the vector of the optimal parameters, which is normally re-scaled and saved into the result file. Alternatively, the user may select the option to repeat the optimisation from a new point nearby the solution which has just been returned. This is achieved by adding a small perturbation to the solution again using random noise. This feature is intended as well to evaluate the consistency of the optimal solution against the possible existence of different local minima. However, it is important here to carefully choose an appropriate magnitude for the random noise. This is because on an optimal trajectory the vehicle is operated on the boundaries of its performance envelope, and even a very small change applied to the optimal control may require the vehicle to operate too far outside its physical limits and may prevent the new optimisation to start.

8.3.5. Running FastLap

The `fastlap` executable program was briefly introduced in §8.1.2. The command line to run the program from the computer console reads:

```
fastlap <data file> <optim. file> <options>
```

where `<data file>` is the name of the file which contains all the necessary data to define an optimisation case, and `<optim. file>` is the name of the file where `fastlap` will save the optimal control and state parameters at the end of the run. These two input arguments are mandatory. Then, a set of pre-defined keywords may be included in the command line, These allow to select among the various program options which are listed below.

- **Optimisation tolerances**; three combinations of values for the major optimality tolerance and the major feasibility tolerance for SNOPT have been pre-set in `fastlap`. The user may select one of them using the following keywords:
  - `-lp`
  - `-np`
  - `-hp`

  which stand for low precision, normal precision and high precision respectively. This option has been used to investigate the trade off between loss of accuracy in the solution and reduced computational time, as will be shown in the next section.
- **Speed of convergence**; this option sets the value of the major step limit parameter for SNOPT and therefore relates to the speed of convergence of the optimisation procedure. The user may choose one of three pre-defined levels, which are identified by the following keywords:
  - `-S0`
  - `-S1`
  - `-S2`

  The first option sets the value of the major step limit equal to 0.01, which usually is a strong limitation to the maximum step size which SNOPT is allowed to take every major iteration. This option is to be used if the problem functions are expected to be very sensitive with respect to some of the independent variables in certain conditions, e.g. when dealing with oversteer biased vehicle set-ups. The second option sets the major step limit equal to 0.1, and is intended as a compromise

between robustness and speed of convergence. Finally, the third option leaves the default setting for the major step limit unchanged.

- **Add noise to initial solution vector**; this option, which was described in the previous section, may be triggered by the user by including the keyword "`-n`".
- **Repeat optimisation**; this option was as well described in the previous section and may be activated by the user by including the keyword "`-r`".
- **Multiple runs**; including a positive integer number among the options forces `fastlap` to repeat the same optimisation case from the beginning as many times as the value of the integer. Every time the optimisation re-starts, some random noise is added to the initial solution vector. Therefore, this option includes automatically "`-n`". Conversely, the option "`-r`" is not available in this case. The optimal solutions for the various runs are saved in separate files, whose names are that of the `<optim. file>` followed by a counter. Finally, running `fastlap` without options is equivalent to:

```
fastlap <data file> <optim. file> -hp -S2
```

8.4. SETTING UP A LAP TIME OPTIMISATION PROBLEM

In this section the necessary steps for setting up a lap time optimisation problem are described. The procedure essentially involves the preparation of the data regarding the track map, the initial vehicle controls and states, and the discretisation scheme. The strategy for refining the discretisation scheme and for fine tuning the optimisation program in order to achieve satisfying accuracy, speed and robustness is also highlighted.

8.4.1. The track data file

All the data which define the optimisation problem are coded in the track data file. The track data file consists of a matrix with eighteen columns. The first column is the independent path co-ordinate s, i.e. the track length measured on the road centre line from the start-finish line. The remaining columns include various information, all coded as functions of the path-co-ordinate s, and precisely the map of the road centre line, the vehicle initial trajectory and controls, the problem discretisation scheme, and the integration step size to be used in each of the track segments. The path co-ordinate s is usually sampled at intervals equal to one or two meters.

Different strategies to obtain circuit maps were presented in §1.5. In the track data file four columns are dedicated to the parameters which describe the road centre line. These parameters are (see figure 8.9):

- The curvature of the road centre line, k_t, or its local radius r_t;
- The tangent angle of the centre line, ψ_t;
- The co-ordinates of its centre line, x_t, y_t, referred to a reference axes system fixed in space whose origin is located at the start-finish line.

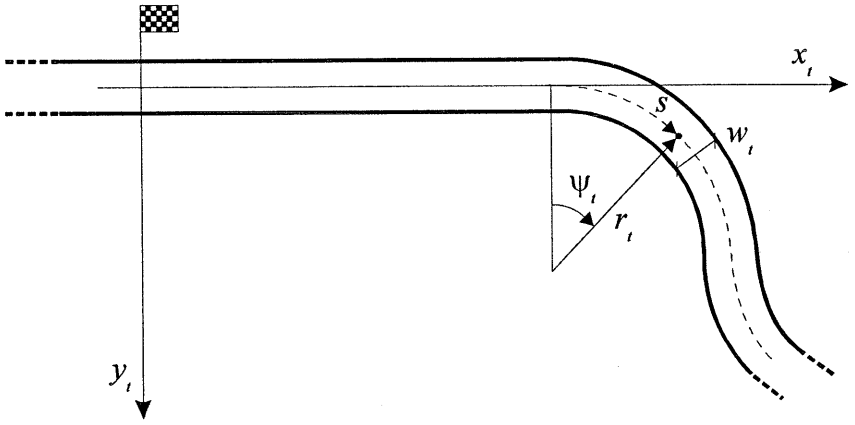


Figure 8.9 Track map description.

Currently, the road width is regarded as constant and road camber and elevation are not included, as the vehicle model implemented in FastLap does not require such information. However, the extension would be straightforward.

Further four columns are used to define the problem discretisation scheme. These informations are coded in the track data file using flags. The flags are integer values associated to the path co-ordinate s, which are normally set equal to zero, unless some action is required at a specified track location. The essential informations for defining the problem discretisation scheme are the following:

- The locations of the junctions between the track segments;
- The locations of the control decision points for the vehicle lateral and longitudinal controls. Two columns of flags are used for the control discretisation grid, since the two control inputs are treated independently. Furthermore, in certain sections of the track it may be convenient to lock the value of the control inputs, e.g. zero steer angle and maximum throttle aperture during straight sections. These different conditions are accounted for by using different values for the control flags;
- The track segments where the road boundary constraints are enforced. These are specified using two different flag values which indicate the beginning and the end of segments where the vehicle distance from the road centre line is to be monitored.

Even though this approach may seem to be rather inefficient, as the four columns are occupied mostly be zero integer values, it is very simple and it allowed easy manual editing of the discretisation scheme during the development of FastLap. General rules for setting up such a discretisation scheme may be derived, as we shall see a little later, and an automated procedure is foreseeable for future versions of FastLap.

An initial vehicle control history and the corresponding vehicle state trajectories are necessary to initialise the independent optimisation parameters at the start of an optimisation run. Eight columns are filled with this information, which include:

- The vehicle lateral and longitudinal controls;
- The vehicle yaw angle;
- The vehicle yaw rate;
- The x and y positions of the vehicle centre of mass;
- The vehicle lateral and longitudinal velocities.

For the independent control parameters $\mathbf{u}_n$, their initial values are extracted from the track data file at the locations specified by the corresponding control flags. For the independent state parameters $\mathbf{x}_m$, some of their values are extracted directly from the track data file at the locations specified by the corresponding segment junction flags, while others are computed from the states specified above. These are the vehicle lateral distance from the road centre line, which may be computed from the x and y co-ordinate of the vehicle centre of mass, and the wheel rotational velocities, which are obtained by dividing the vehicle longitudinal velocity by the wheel radii.

Finally, one last column contains the integration step size for each of the segments. Even though within each segment the vehicle dynamics equations are numerically solved using a fixed-step method, specifying different step size values for different segments of the track allows to reduce the computational time significantly. Essentially, the slow segments are the bottleneck for the simulation. When large driving or braking torque is applied to the wheels compared to the longitudinal tyre forces available, it is more difficult to obtain sufficient accuracy with regard to the wheel spin equations. Conversely, in very fast sections, e.g. long straights, sufficient accuracy can be achieved using integration steps ten times larger than for the slower sections. This simple strategy for adapting the integration step size based on pre-determined values assigned along the track allows a considerable increase in the simulation and optimisation speeds and avoids some of the complications involved in applying Automatic Differentiation when using a variable-step numerical integration routine (Eberhard and Bischof, 1999).

8.4.2. Getting the initial vehicle controls and state trajectories

The first step for getting the initial vehicle controls and state trajectories is to obtain an approximated racing line. In chapter 3 a method for the reconstruction of the racing line using measured data from the real car was devised. Such method would be ideal to start with. However, the reconstructed racing line does not usually match the corresponding circuit map closely enough, as was shown in chapter 3, to be used here without further adjustments. Therefore, a completely different approach was used. That is, the initial racing line is simply "drawn" over the map of the circuit. A Matlab program was written which allows the user to select points on the map of the circuit using the mouse, pointing where the racing line ought to be. Then, the racing line is obtained using spline interpolation techniques. An example is shown in figure 8.10.

Next, it is necessary to define the initial vehicle velocity profile along the track. In general, one would be inclined to specify a velocity profile as close as possible to the vehicle performance limits, so that the optimisation process would take less iterations. However, since the aim is to derive an initial set of controls which is feasible for all the anticipated vehicle configurations, it is convenient to sacrifice some extra time for the optimisation to compute, and remain on the safe side. Hence, the strategy which is applied here is to start from the velocity recorded on the track from the real car during a generic lap in race trim and then reduce it by as much as 25%.

Finally, the driver model which was described in chapter 4 is used to derive the initial vehicle controls which correspond to the desired initial manoeuvre. Here, the methodology for the implementation and use of the multiple preview control driver model which was outlined earlier fully applies, with only one exception. The time to distance scaling factor S_{CF} is usually related to the reference line where the path

distance is measured. In the path following trials which were solved in chapter 5 the time to distance scaling factor referred to the same path that the driver model tried to follow. Here, while the driver model looks ahead at the racing line, the time to distance scaling factor must be computed with respect to the road centre line. This is because the road centre line is where the path distance is measured, and the vehicle control input must be a function of the same travelled distance.

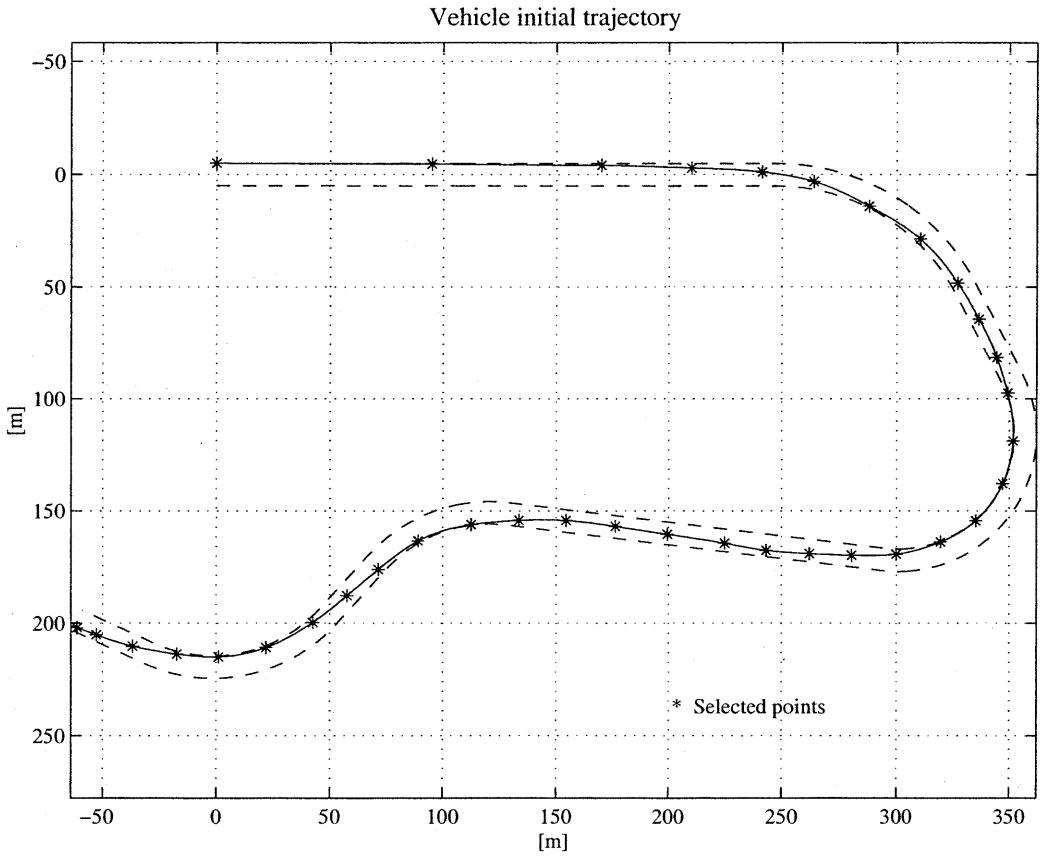


Figure 8.10 Vehicle initial trajectory from manually selected points.

8.4.3. General rules for setting up the discretisation scheme

The discretisation scheme for the vehicle lateral and longitudinal controls relates to the accuracy of the solution. As was discussed in chapter 2 and 6, direct methods for the solution of optimal control problems only return approximated solutions. This is because the original continuous control history is replaced with a discrete approximation which has a finite number of degrees of freedom. Hence, the general rule is that increasing the number of control decision points improves such approximation. From this point of view, using Automatic Differentiation is a great advantage. Since the computational time to evaluate derivatives with AD is nearly insensitive to the number of independent variables, maximum freedom is allowed in refining the discretisation of the control history. However, using a very fine control discretisation grid with many control decision points evenly spaced over the entire track length would not be the ideal strategy, as it would increase the size of the optimisation problem and its number of

degrees of freedom[1]. This may potentially slow down the optimisation by increasing the number of iterations required to reach the solution. Instead, it will be more convenient to place more control decision points where the rate of change of the controls is expected to be greater, e.g. at braking points and during corners. Furthermore, despite using AD, the accuracy of the derivatives for the lap time optimisation program is strongly limited by the accuracy of the numerical solution of the vehicle dynamics equations. If the control intervals are chosen to be very short, each control decision point affects only a very short portion of the vehicle trajectory as well, and the magnitude of the derivatives will consequently become very small and ultimately will be badly affected by the numerical noise induced by the integration of the vehicle differential equations. Hence, there will be a minimum length for the instances of the control discretisation grid, below which it is not advantageous to go.

The discretisation of the vehicle state trajectories relates to the robustness of the lap time optimisation program, as was extensively discussed in §6.3. Sensitivity problems arise essentially when the rear tyres of a vehicle saturate before the front ones. Hence, it is good practice to devise the optimal state discretisation scheme using oversteer biased vehicle configurations. When sensitivity problems occur, the remedy is to shorten the length of the trajectory segments where such problems are found. Yet, the ideal scheme does not require that all the segments be of equal length. In fact, the length of the segments is related to the geometry of the track and the vehicle speed. Slow and twisty sections require the shortest lengths for the trajectory segments, while medium and fast corners allow longer segments. Finally, increasing the number of trajectory segments increases as well the size of the optimisation problem, but not its number of degrees of freedom, as each set of independent state parameters at each junction is related to an equivalent set of continuity constraints. Therefore the efficiency of the optimisation is not significantly penalised.

A particular problem discretisation scheme has been devised for FastLap. It allows sufficient freedom for adapting the scheme to the different conditions which may arise in different parts of the circuit, but at the same time is easy to implement and upgrade, allows to determine the sparsity pattern of the Jacobian easily by means of an automated procedure, as was anticipated in §8.3.3, and allows fast interpolation for the vehicle control inputs. The key idea is the definition of a *base* grid of points, which are evenly spaced with their separation distance representing the minimum length allowed for the control intervals. The instances in the control discretisation grid may differ from each other, but they are always a finite multiple of the base interval. Furthermore, each trajectory segment always includes a finite number of control intervals. Hence, there will always be a control decision point located at the junction between two segments, and the length of each segment will also be an integer multiple of the base interval.

The main advantage of this strategy is that it allows fast interpolation of the vehicle control inputs. The first step in an interpolation algorithm is to find the index in the table which is nearest to the input value, whose corresponding output is to be computed. In a general case where the elements in the table are positioned at random distances from each other, it is necessary to perform a search over the whole table which is usually computationally expensive. Conversely, if the elements are equally spaced, the index may be computed with a simple algebraic operation, as was described in

[1] The number of degrees of freedom in a constrained optimisation problem is defined as the difference between the number of independent variables and the number of non-linear constraints which are active at the solution.

§7.3.4.1. In FastLap the base grid serves exactly this purpose. Every time the actual control input has to be evaluated by means of interpolation, the index of the nearest base point is firstly computed. However, since the base grid is associated to control discretisation grid, the nearest control point is also immediately available, without requiring further effort.

In summary, the lap time optimisation problem in FastLap is characterised by three discretisation grids, as is qualitatively illustrated in figure 8.11, based on three different distance scales:

- The trajectory segments, with the largest distance scale, which includes several control intervals;
- The control intervals, characterised by a medium distance scale, with each interval equal to a finite multiple of the base interval;
- The integration step size, with the smallest size, which is constant within a trajectory segment, but may vary from one segment to another.

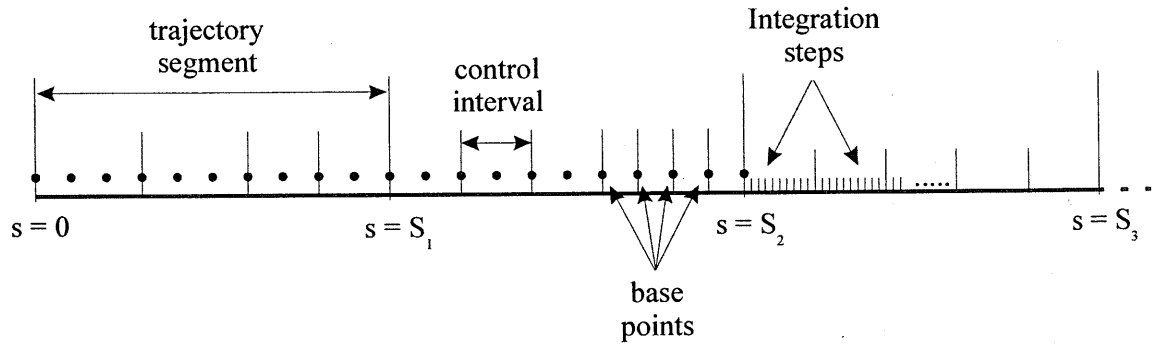


Figure 8.11 Distance scales for the trajectory and control discretisation grids, and integration steps.

8.5. CONCLUSIONS

The implementation of the lap time optimisation program FastLap has been described. General rules for setting up an optimisation case have been highlighted. At present, setting up an optimisation case requires a considerable amount of manual work, particularly for deriving the initial trial solution and for defining the discretisation scheme. This approach is convenient during the development phase of FastLap, where the many problem parameters have to be modified frequently in order to refine the solution. However, more automation is desirable when thinking of the possible applications of FastLap to analyse practical problems.

First of all, we have observed that the characterisation of the discretisation scheme relates to the geometry of the circuit and the typical velocities with which the racing car can traverse a lap. It appears therefore possible that heuristic rules can be derived and implemented in an automated algorithm for generating the discretisation scheme for a given circuit. Furthermore, the strategy of defining the distance scales as multiples of the same base value, allows to refine the discretisation for the whole problem by

varying a single parameter.

In the context of the work which is normally required to obtain the digital map of a circuit, deriving the initial estimate of the racing line by using the spline interpolation technique described earlier does not add a considerable amount of work. Besides, it only needs to be done once for a circuit. The automated reconstruction of the racing line from telemetry data certainly represents a better option, but further work is needed to enhance its accuracy so that the reconstructed racing line is not too far off the intended track.

Many are the parameters that one needs to decide and tune in order to solve a lap time optimisation problem efficiently, and at first the process would seem to be endless. Even though we have highlighted in a general way the rules to get started, many questions remain, such as how long should the trajectory segment be, or how many control decision points are effectively needed, and so on. In the following section, we shall apply FastLap to find the solutions for a variety of minimum time vehicle manoeuvring problems, and we shall to answer all the remaining questions.

Chapter 9.

Optimisation Results

In this chapter FastLap is used to solve various minimum time vehicle manoeuvring problems. Firstly, the general framework to define the problem parameters and to analyse the results, with particular emphasis on accuracy and reliability, is presented. Optimal solutions are obtained for two current Formula One circuits, Suzuka and Barcelona. Furthermore, a special double lane change manoeuvre has been designed for the purpose of special analyses, such as the yaw sensitivity analysis. Most of the results have been obtained using realistic vehicle parameters for a typical 1999 Formula One car in race trim configuration. Also, a radically different set of parameters, resembling a rear wheel drive rally car operating on a low-friction road surface, e.g. gravel, has been implemented and used for comparison of the typical handling behaviour of Formula One cars vs. rally cars. The optimal solutions obtained using these default sets of parameters are firstly presented and discussed. Then, the sensitivity of the performance of the Formula One car to the variation of some design parameters is investigated, particularly with regard to the vehicle yaw inertia, the vehicle total mass and the vehicle mass distribution.

9.1. THE GENERAL FRAMEWORK

Deciding the various lap time optimisation problem parameters is essentially an iterative procedure. One has to start with a trial set of values and see whether a satisfactory solution can be achieved. If this is not the case, one needs to apply appropriate changes to the initial set of problem parameters and repeat the optimisation run. The procedure is not straightforward and usually requires a few trials before the proper problem set-up is defined. However, this needs not to be done every time, but only when a new circuit or a new vehicle model is added to the library of the optimisation program.

To start with, we shall assume that the track map is given and that the initial vehicle trajectory and controls have been obtained as was described in the previous chapter. Then, the problem parameters which have to be chosen are:

- The integration step size;
- The control discretisation grid;
- The trajectory discretisation grid;
- The optimisation tolerances.

A suitable value for the integration step size is firstly identified by evaluating the objective function and the problem derivatives for decreasing values of the step size,

until sufficiently accurate results are obtained. The procedure was already illustrated in the example of §7.4, where the values of the objective function J and its directional gradient were represented as functions of the integration step. For large lap time optimisation problems, though, it is convenient to identify different integration steps for different track segments, as it was observed that fast track sections require a significantly larger step size compared to slow ones. Hence, the identification of the step size has to be performed separately for different track sections. The accuracy of the objective function should be sufficiently greater than the desired accuracy of the optimal solution. For FastLap the goal is to evaluate the minimum lap time with the best accuracy of the order of one millisecond. Therefore, an absolute accuracy for the objective function of the order of 1e-004 [s] is regarded as sufficiently safe to start with.

Next, the control and trajectory discretisation grids have to be decided. An initial estimate of the length of the control intervals may be derived by observing that real racing drivers apply a control input to the car whose bandwidth is of the order of 2 to 4 Hz. Hence, it appears reasonable that an acceptable initial control discretisation grid consists of control decision points whose relative distance is of the order of the distance travelled by the car in about 0.25 [s], i.e. 5 to 25 [m]. With regard to the trajectory discretisation grid, one may observe that in order to achieve effective de-coupling between early controls and later vehicle states, the length of a trajectory segment should not exceed the distance that the vehicle travels during a transient phase which results from a control input variation applied at the beginning of the segment. This idea leads to the correlation of the length of the trajectory segments with the preview distance for the driver model which was discussed in chapter 4. Hence, we shall assume as initial segment length the distance travelled by the vehicle in one second, i.e. 20 to 100 [m].

Finally, the feasibility and optimality tolerances for the optimisation algorithm SNOPT have to be set. As was briefly described in the previous chapter, these tolerances relate to the reduced gradients and not directly to the values of the objective and constraint functions. In other words, setting the optimality tolerance equal to 1e-003 does not imply an accuracy of the order of one millisecond when the optimisation converges. Furthermore, the best tolerances which can be met at the solution depend on the accuracy of the objective and constraint functions and of their derivatives, which, in turn, depends on the integration step chosen. This cannot be predicted with total confidence, since the numerical integration algorithm uses a fixed step. Therefore the appropriate strategy is to start with rather large values for these tolerances, e.g. both feasibility and optimality tolerances equal to 1e-002, and progressively refine the accuracy of the solution together with the other problem parameters.

Having decided all the problem parameters, the first lap time optimisation may start. It is very unlikely that the initial problem set-up leads to a satisfactory solution, and even if the optimisation converges nicely to *a solution*, further work is necessary to ensure that such solution is truly the optimal one. Various situations may arise, and, in order to decide how to intervene, the report file which SNOPT generates during an optimisation run is extremely useful. Particularly, SNOPT writes an iteration log which includes, among other information, the step length τ taken along the current search direction, the value of the merit function, the measures for the feasibility and optimality criteria, and the condition number of the Hessian matrix of the Lagrangian function. Furthermore, at the end of the optimisation, SNOPT prints the values of all the dependent variables and their gradient and reduced gradient components.

In summary, the more or less troublesome situations which may occur are

described in the following list.

- **Optimisation converged successfully**. This is the message which we obviously very much hope for. However, it does not mean that everything is already working well. Examining the optimisation report reveals useful information on how the whole optimisation went. Firstly, a good sign is if the step length τ assumes values equal to 1.0 regularly towards the end of the optimisation. This is expected when approaching the solution, as the norm of the search direction vector should decrease steadily. Secondly, the feasibility and optimality criteria should decrease regularly until they meet the specified tolerances, and also the value of the condition number of the Hessian should remain low throughout the whole optimisation, e.g. not greater than 1.0e5. Finally, as was anticipated in chapter 8, the maximum number of major iterations allowed is limited to 200, in order to contain the time to obtain a solution within practical values. Hence, if the optimisation took about 100 iterations, we may assume that the general setting is acceptable also in terms of tolerances. Instead, if the solution was obtained in few iterations, say 20 to 50, there is certainly a case for repeating the optimisation with tighter tolerances.
- **Optimisation does not converge**, the feasibility and optimality criteria are very close to their tolerance values, but the optimisation program keeps iterating. This is usually a clear indication that the accuracy of the objective and constraint functions and their derivatives is not sufficient in relation to the optimisation tolerances required. The option is either to enlarge these tolerances or decrease the integration step size in the numerical simulation of the vehicle dynamics equations.
- **Optimisation does not converge**, the feasibility and optimality criteria remain very large, and usually increase suddenly at some major iterations. This is usually accompanied by large condition numbers of the Hessian matrix and very small line search step lengths, and is a clear indication of sensitivity problems. The solution is to reduce the length of the trajectory segments, possibly in the section of the track where it is needed. In fact, it may not be necessary to update the whole trajectory discretisation scheme, which would increase the problem size more than is necessary. The optimisation report saves the values of the gradient components of the objective function with respect to all the independent variables at the last major iteration. It is a time consuming job to inspect the report when there are thousands of variables, but it is usually possible to find which gradient component is a few orders of magnitude greater than the average, and to work out where on the track the corresponding control or state optimisation parameter occurs. This is important as in the early stage of the development it is convenient to optimise the discretisation strategy for the various geometric features of a circuit, and the experience gained enables to derive general rules which may be applied when modelling new circuits.
- **Optimisation fails to start**. This indicates bad errors in computing the objective and constraint functions and, more likely, their derivatives. It was observed in chapter 7 that for relatively large values of the integration step size, while the objective and constraints are still computed to a reasonable accuracy, the errors in the derivatives tend to rise much more rapidly, see §7.4.2, and if the integration step size has been largely overestimated, the values for the derivatives may be meaningless and may prevent the start of the optimisation.

When regular convergence is achieved, as is described in the first of the above cases, a further phase of development may begin. Firstly, it is necessary to refine the control discretisation grid by adding more control decision points, and see whether the optimal solution improves further, until no significant improvement is gained. Then, it is necessary to evaluate the same optimisation problem from different initial sets of optimisation parameters. This reveals the presence of local minima and gives the final indication of the actual accuracy which the lap time optimisation program yields.

9.2. INTRODUCTION TO THE RESULTS

The aim of this chapter is to demonstrate the accuracy and the reliability of the results obtained from FastLap. We shall assume that all the problem parameters, which have been derived applying the methodology described above, are fixed. In order to investigate the accuracy of the optimal solution, each optimisation case is solved repeatedly, usually 4 to 6 times, by perturbing the initial guess of the solution. Furthermore, FastLap allows three levels of optimisation tolerances. All the cases have been solved using the highest level of accuracy. However, by inspecting the optimisation report it has been possible to derive the value of the objective function, i.e. the lap time, and the number of iterations corresponding to the two lower levels of accuracy. This will be used in order to give some indication of the trade off between accuracy and computational time which characterises the current implementation of the lap time optimisation program.

In the next tables the various problem parameters are summarised. Table 9.1 shows the settings for the feasibility and optimality tolerances in SNOPT which correspond to the high, normal and low precision options for FastLap. Table 9.2 reports the parameters describing the problem discretisation scheme for the four tracks which are used here, i.e. the Suzuka and Barcelona circuits, and two versions of the double lane change manoeuvre. Furthermore, figure 9.1 shows as example the trajectory discretisation scheme for Suzuka. The colour coded dots mark the junctions between the segments, and the colour relates to the integration step size for that segment. The slowest parts of the track feature very short segment with the smallest integration step, while the straight sections allow much longer segments and larger integration steps. Progressive transition is provided between different grid sizes. Finally, all the optimisation runs have been performed on two different UNIX workstations, whose details are reported in table 9.3.

OPTIMISATION TOLERANCES IN FASTLAP			
	Low precision	**Normal precision**	**High precision**
Feasibility Tolerance	1.0e-3	5.0e-4	5.0e-4
Optimality Tolerance	1.0e-2	7.5e-3	5.0e-3

Table 9.1 Tolerance settings.

PROBLEM DISCRETISATION PARAMETERS				
	Suzuka	**Barcelona**	**Double Lane Change**	**Long Double Lane Change**
Track length	5864 m	4728 m	300 m	396 m
Road width	10 m	10 m	10 m	10 m
Number of trajectory segments	279	242	22	26
Trajectory segment lengths	8 to 100 m	12 to 100 m	12 to 24 m	12 to 24 m
Integration step size range	0.01 to 0.1 m	0.025 to 0.1 m	0.025 to 0.05 m	0.025 to 0.05 m
Number of steer control nodes	919	834	72	84
Number of throttle or brake control nodes	932	849	74	94
Control intervals range	4 to 20 m	4 to 20 m	4 to 8 m	4 to 8 m
Number of independent variables	4362	3861	335	403
Number of constraints	2725	2353	211	247

Table 9.2 Problem discretisation and track parameters.

WORKSTATIONS TECHNICAL DETAILS		
	Compaq	Sun
CPU	Alpha Digital EV6	Ultra Spark II
Number of CPUs	1	6
Clock speed	700 MHz	336 MHz
Memory	512 Mb	4096 Mb
Operative system	Digital UNIX Ver. 4.0E	Sun OS v 5.6 Solaris 2.6
Compiler	GCC version 2.95.2 (C/C++ and Fortran)	CC Workshop compiler ver. 4.2 (C/C++ and Fortran)

Table 9.3 Technical details of the UNIX workstations employed.

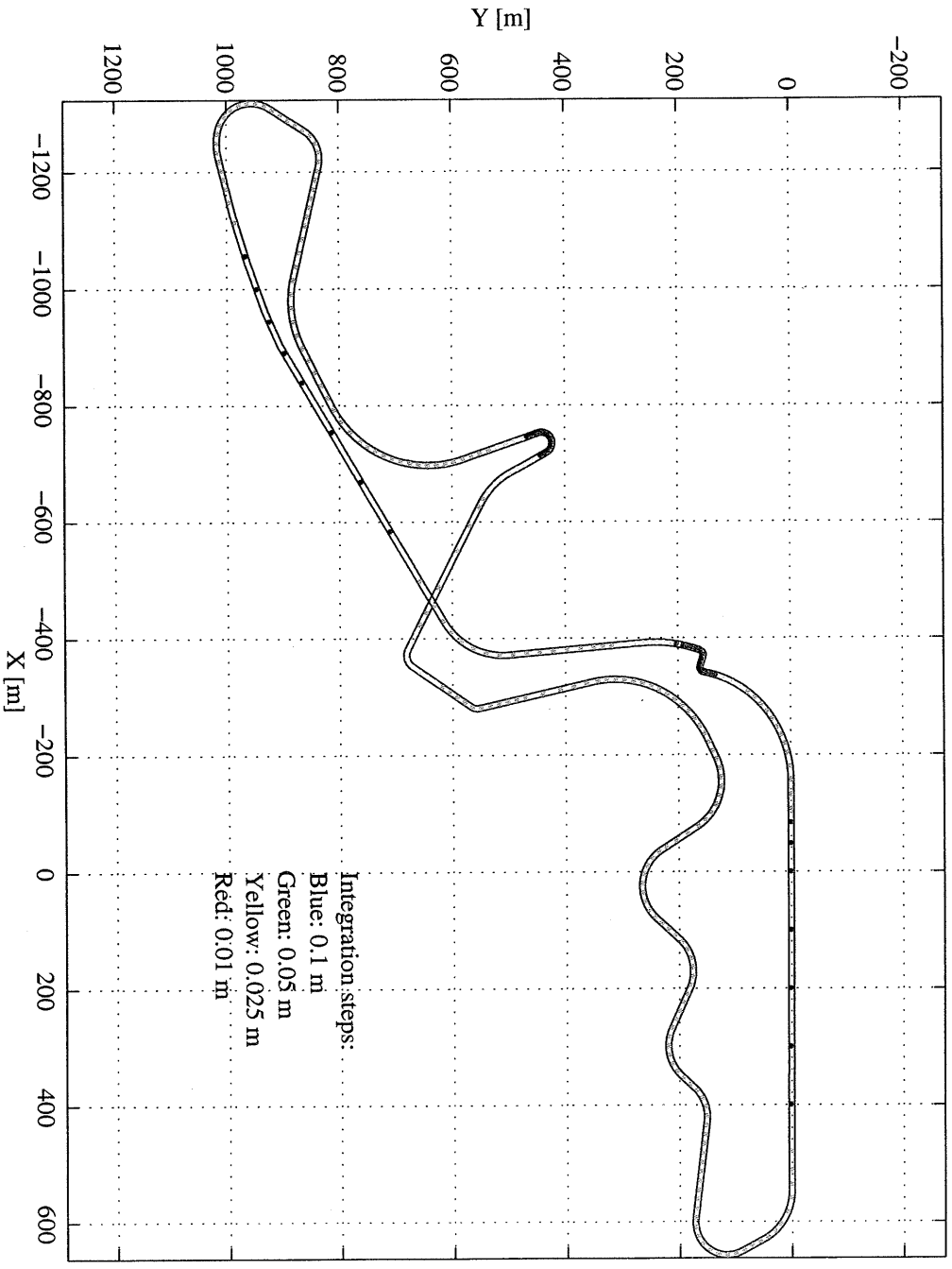


Figure 9.1 Trajectory discretisation scheme for Suzuka, and local integration step size.

9.3. CIRCUIT LAP TIME OPTIMISATION

The first set of results presented in this section shows the optimal trajectories computed for two current Formula One circuits, Barcelona and Suzuka, using the vehicle model with the nominal race trim configurations. The graphs relate to the best solution obtained out of four optimisation runs for each circuit, using FastLap with the default high precision tolerances. The lap times corresponding to the initial solution guesses were 1' 37" for Barcelona and 2' 13" for Suzuka. Table 9.4 summarises the lap times and the number of iterations for each case. The difference between the best and worst minimum lap times is for both circuits of the order one and a half tenths of a second. While all the four runs for Suzuka converged nicely in less than one hundred iterations, only one of the runs for Barcelona converged to the specified tolerances, while the others reached the maximum limit of 200 major iterations imposed. The results were accepted anyway, as the feasibility and optimality criteria were very close to the tolerances. This fact, though, may be indicative that the trajectory discretisation for Barcelona needs further refinements. Finally, the optimisation for Suzuka was run on the Alpha Digital workstation and the average time to compute one iteration was 6.5 minutes, which yields a total computational time of 7 to 10 hours for each of the four cases. The optimisation for Barcelona was run on the Sun Spark workstation, whose single processor speed is significantly lower compared to the Alpha Digital's, and which requires an average time of 18 minutes to compute one iteration, leading to a total time of 28 hours for the run number 3 and 60 hours for the others. However, up to six cases could be run in parallel, giving an overall performance equal or even greater than for the single processor Alpha Digital workstation.

LAP TIME OPTIMISATION SUMMARY				
	Barcelona		**Suzuka**	
	Lap time	Num. of iter.	Lap time	Num. of iter.
Run N. 1	**1' 20" 714**	200	1'36" 206	92
Run N. 2	1' 20" 848	200	1' 36" 247	70
Run N. 3	1' 20" 871	93	**1' 36" 145**	99
Run N. 4	1' 20" 735	200	1' 36" 305	81

Table 9.4 Circuit lap time optimisation results summary.

Compared to real lap times registered during the 1999 season in Suzuka and Barcelona, the results computed by FastLap are significantly lower. For example, in race trim configuration the best lap times recorded in Barcelona and Suzuka were of the order of 1' 25" and 1' 41" respectively. This large difference, though, is not unexpected. Firstly, the tyre forces given by the mathematical tyre model are measured in laboratory conditions, which will certainly differ from track conditions. As a result, the theoretical forces often tend to overestimate the real ones. Secondly, the theoretical results are sensitive to small errors in modelling the track. For example, a small error in the curvature for a corner which precedes a long straight may lead to significantly different speeds over a large section of the track. Furthermore, the track model is perfectly flat. Finally, the optimiser is capable to "drive" the car on the very limit for the whole lap, and it will use the car natural dynamics to its advantage in a way which may not be possible even for the most skilled professional driver.

9.3.1. The optimal trajectory

Figures 9.2 and 9.14 show the computed optimal vehicle trajectories for Barcelona and Suzuka respectively. Realistic racing lines are found for the various track geometries, from slow corners such as hairpins and chicanes, as well as medium speed corners and fast bends. Furthermore, the optimisation program finds straight trajectories which minimise the travelled distance along straight track segments, without the need of imposing road boundary constraints.

9.3.2. Vehicle controls

Figures 9.3 and 9.15 show the optimal steer angle control input in comparison with a typical recording from the real car (both sets of data for Barcelona and Suzuka were taken from the warm-up sessions of the 1999 Grand Prix season). No attempts were made to adjust the vehicle parameters and the tyre friction limit in order to fit the measured data and reproduce the real lap time. The comparison between the theoretical and measured steer angle control is intended purely to highlight some of the features of the optimal solution.

The computed steer angle input features high frequency noise in certain areas, with very large peaks and rapid variations, particularly where the car brakes very heavily before slow corners. It is believed that the noise is a consequence of the fact that linear interpolation is used for the control input and the rate of variation of the steer angle is not limited. Furthermore, when the vehicle is operated on the limit of its tyres under braking, the lateral force available is very small and rather insensitive to variations of the slip angles. Comparing the solutions for the two circuits further validates the above considerations. For example, the first part of Suzuka, e.g. from 0 to about 2000 [m], features a sequence of medium speed corners with very light braking between them. For this section of track, the computed steer angle input compares qualitatively well with the measured one, without too much noise. Conversely, for Barcelona, the steer angle is very noisy in those areas where maximum braking is applied at the end of fast sections, e.g. at s (travelled distance) equal to 800[m] (before turn 1), 1700 [m] (before turn 4), 2500 [m] (before turn 7) and 3500 [m] (before turn 10).

Besides the high frequency noise discussed above, the computed steer angle appears to be far less smooth than the measured one, giving the impression that either the vehicle model is very difficult to control on the limit, or that the optimiser operates the vehicle with a control bandwidth which is wider than that of a professional driver. In order to investigate the last point, an FFT analysis for computed and measured steer angle controls was made and is presented in figures 9.4 and 9.16. Such analysis, however, does not reveal a significant difference, as most of the control activity is concentrated at low frequencies for both theoretical and measured steer controls, with the computed steer angle only having marginally greater amplitudes at frequencies higher than 0.5 [Hz]. Nevertheless, the results indicate that the vehicle model is very difficult to control on the limit, and the optimiser has control capabilities beyond those of a real driver.

Finally, figures 9.5 and 9.17 show the computed throttle/brake control input. The advantage of the control discretisation is evident here, as it allows to model effectively a control input which is nearly discontinuous. In braking, the control reaches often the maximum value allowed. This is unusual, since the tyre limit should be reached first,

and it indicates that the value of the maximum braking torque specified is too low.

9.3.3. Vehicle speeds and attitude

The subsequent four graphs for each circuit show the vehicle speeds and attitude (in terms of the difference between the front and rear slip angles) along the optimal lap. The vehicle yaw rate appears to be somewhat noisy, with sharp peaks often occurring at the beginning of some turns, probably as a consequence of the noise in the steer control input, but also as a result of the fact that the current vehicle model uses steady state approximation of the wheel load transfer as well as steady state tyre forces. Furthermore, the fact that the tyre forces respond with no delay to a control input variation may be one more cause of the over optimistic results yielded by FastLap.

Figures 9.7 and 9.19 show the computed vehicle longitudinal velocity in comparison with a typical measured velocity. The two graphs compare actually rather well in terms of low speed corners and acceleration rates. However, big gaps exist sometimes at high speed corners, and such gaps may affect long subsequent track sections. One example is the speed at turn 9 in Barcelona, at s equal to 2900 [m], where the computed speed is about 7 [m/s] higher than the measured one, and affects the velocity for the whole of the next straight section. Such a large difference may be partially due to inaccuracy in the track model. However, figure 9.8 reveals the possibility of a different explanation. The vehicle lateral velocity before turn 9 peaks to a value as large as -14 [m/s], which indicates an oversteer attitude in entering the right hand corner, also confirmed by figure 9.9 which shows the difference between front and rear slip angles. It seems as if the optimisation algorithm had found a driving strategy more typical of a rally car which enables much greater speed through the corner.

For both circuit lap optimisations, the vehicle lateral velocity and slip angle difference indicate this particular vehicle behaviour, which would be certainly less than ideal, where the car can have either an understeer attitude and a moment later change to an oversteer attitude. This seems to be due to the fact that both front and rear axles reach their lateral limit nearly at the same time, as will be discussed next. The optimiser seems to be able to take advantage of this behaviour, but the nature of the solution suggests that such a vehicle would be very hard to control.

9.3.4. Vehicle acceleration and performance limit

Figures 9.10 and 9.22 show the g-g diagram for the two optimal laps in Barcelona and Suzuka respectively. These graphs give an indication of the maximum values of lateral and longitudinal acceleration which the vehicle model is capable to achieve. Figures 9.11 and 9.23 show also the acceleration vector of the vehicle centre of gravity along the optimal trajectory. This shows nicely how the optimal driving strategy is a continuous combination of braking, turning and driving, much in the way that was described in the introduction in §1.2.

Finally, figures 9.12 and 9.13 for Barcelona and 9.24 and 9.25 for Suzuka show the tyre saturation level for each individual tyre in both lateral and longitudinal directions (see Appendix A for the definition of the tyre saturation level). The front tyres are consistently used up to 100 % of their capabilities in cornering, and also the rear tyres frequently reach their lateral limit. This confirms the scenario which is described above, with the vehicle being in balance between understeer and oversteer, leading to a rather unpredictable behaviour.

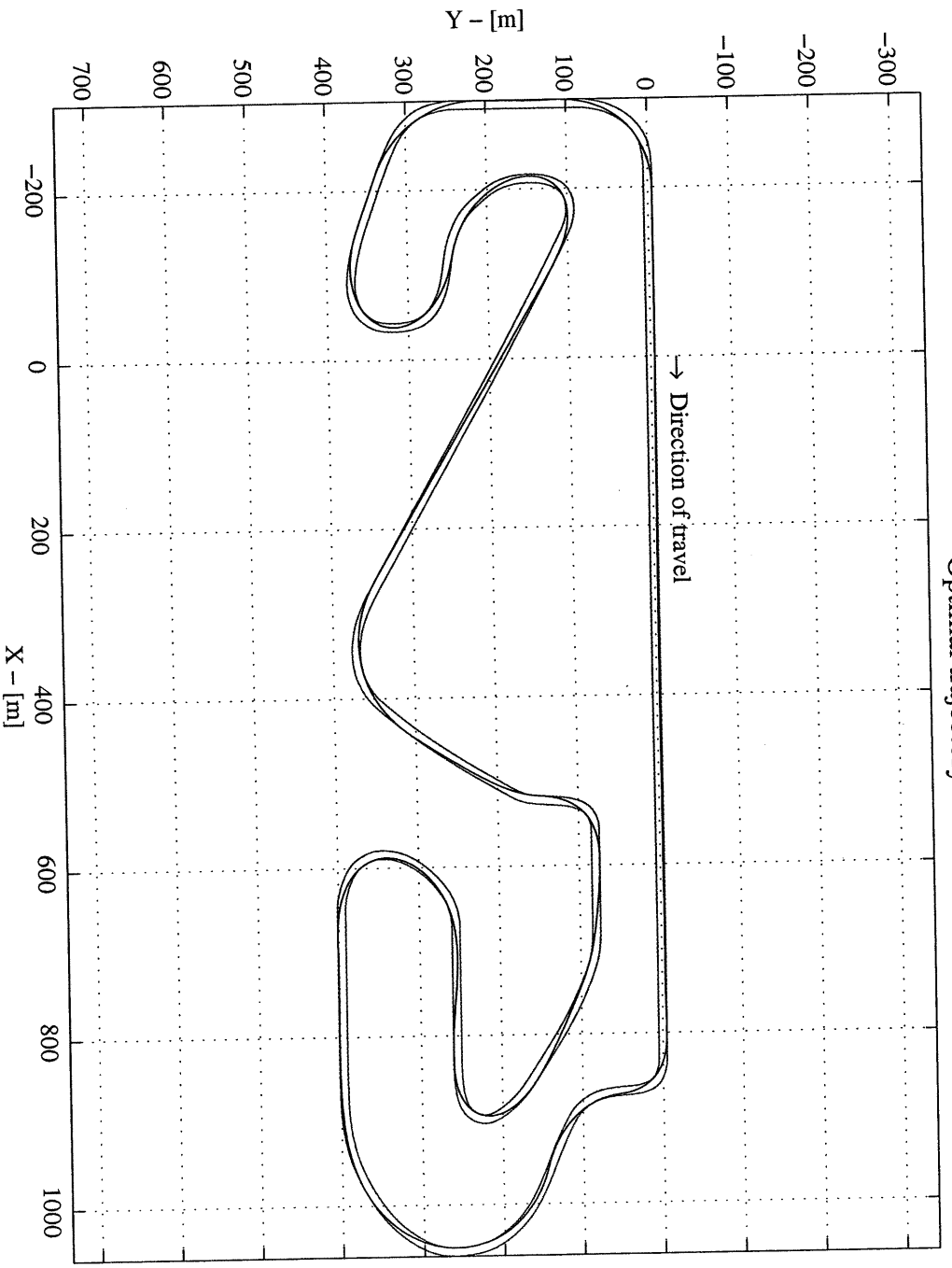


Figure 9.2 Optimal vehicle trajectory, Barcelona circuit.

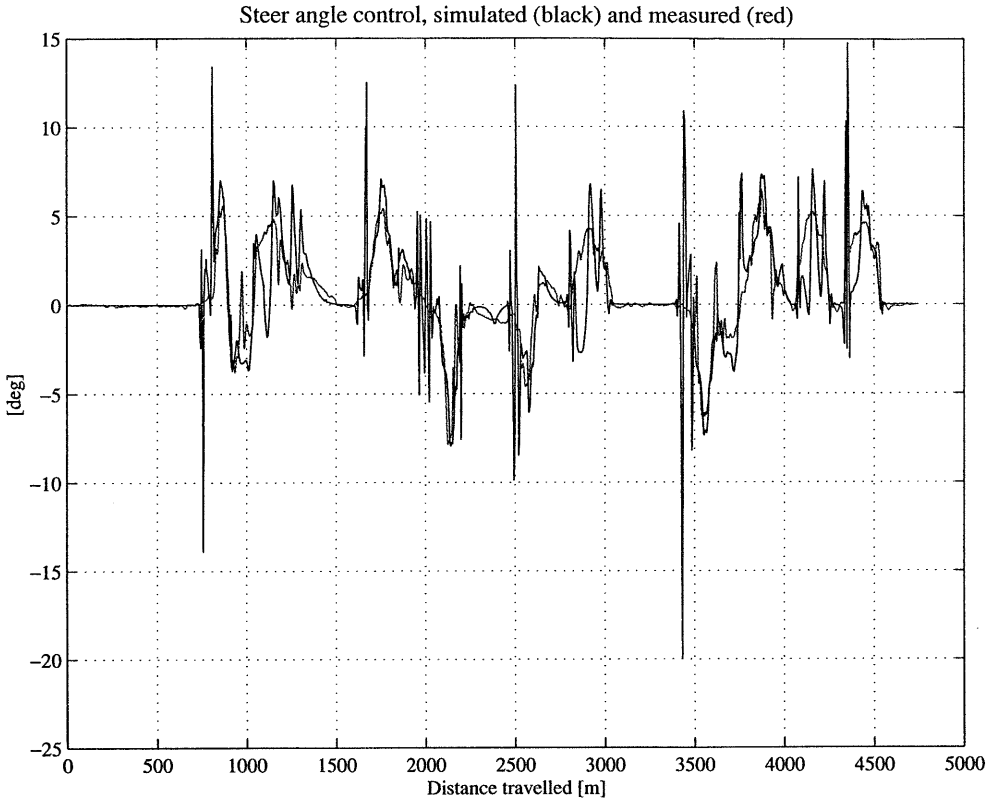


Figure 9.3 Computed and measured steer angle, Barcelona.

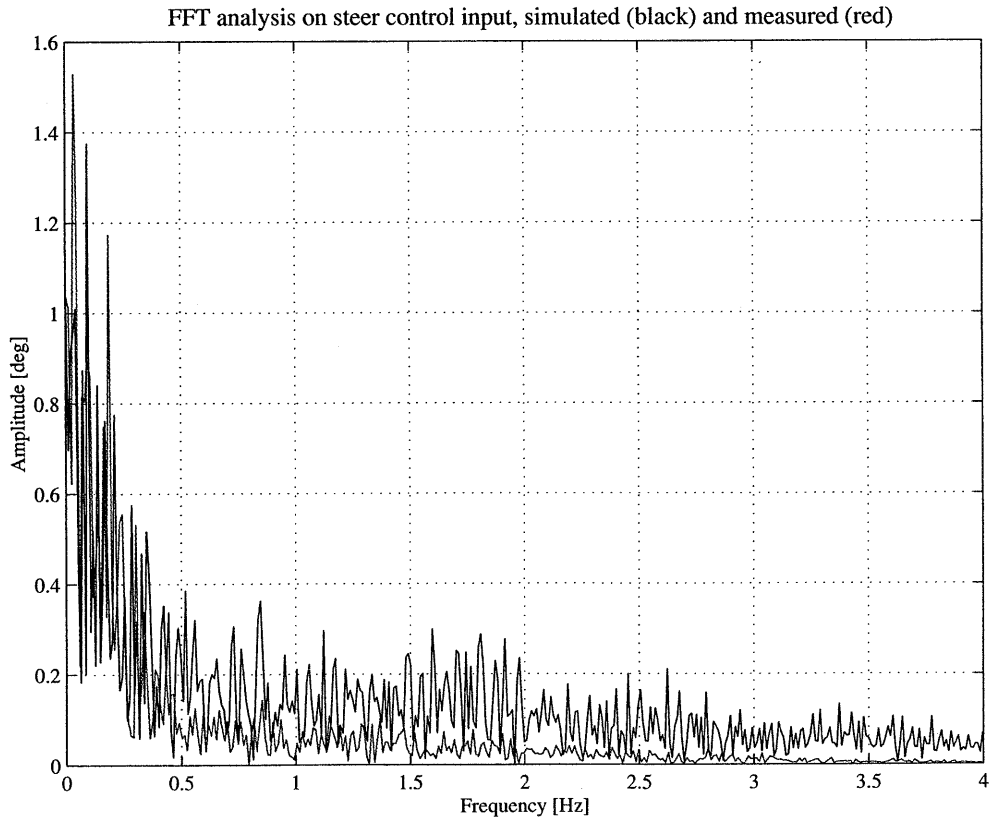


Figure 9.4 FFT analysis of computed and measured steer angle, Barcelona.

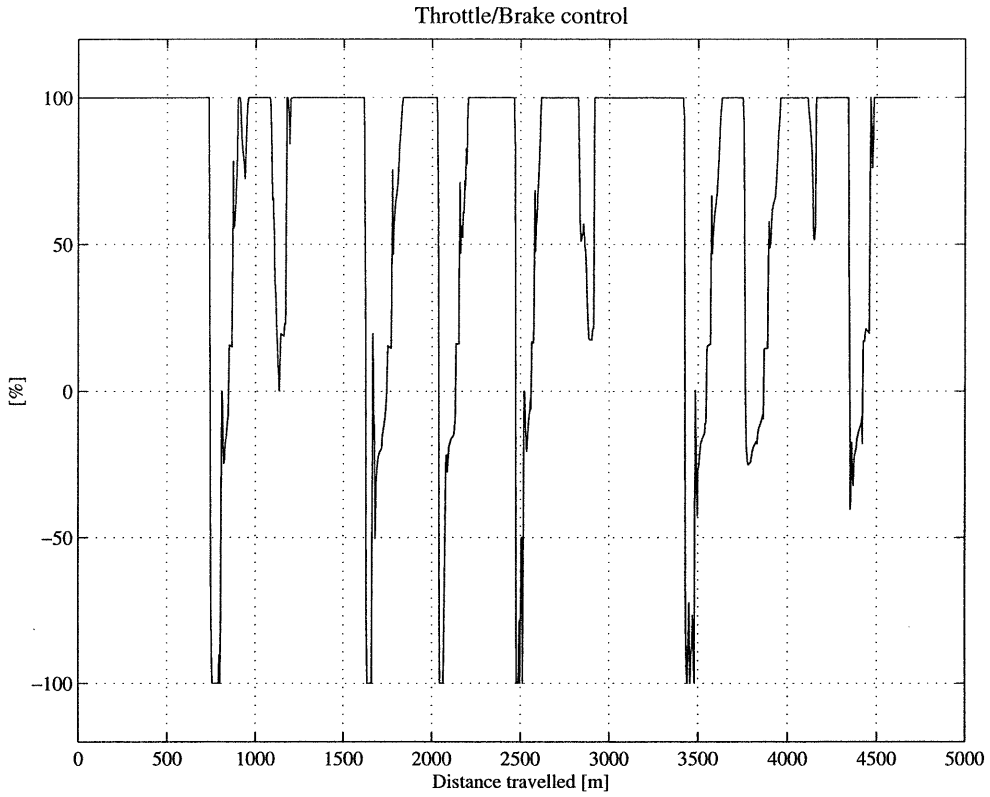


Figure 9.5 Throttle and brake vehicle control, Barcelona.

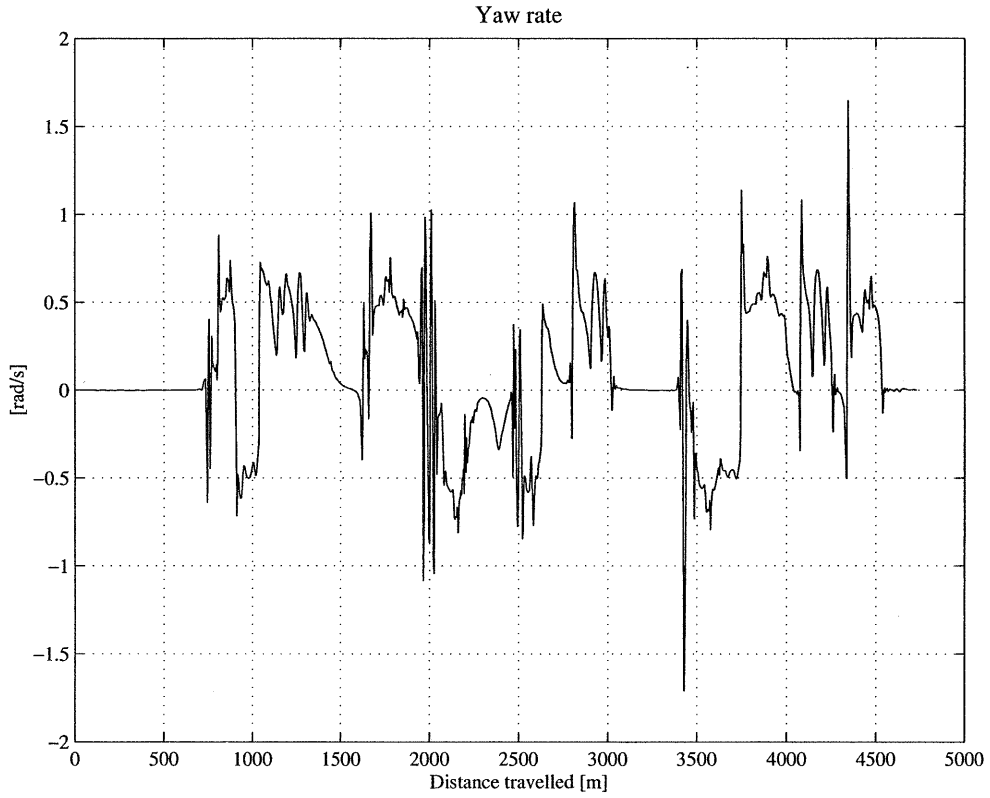


Figure 9.6 Vehicle yaw rate, Barcelona.

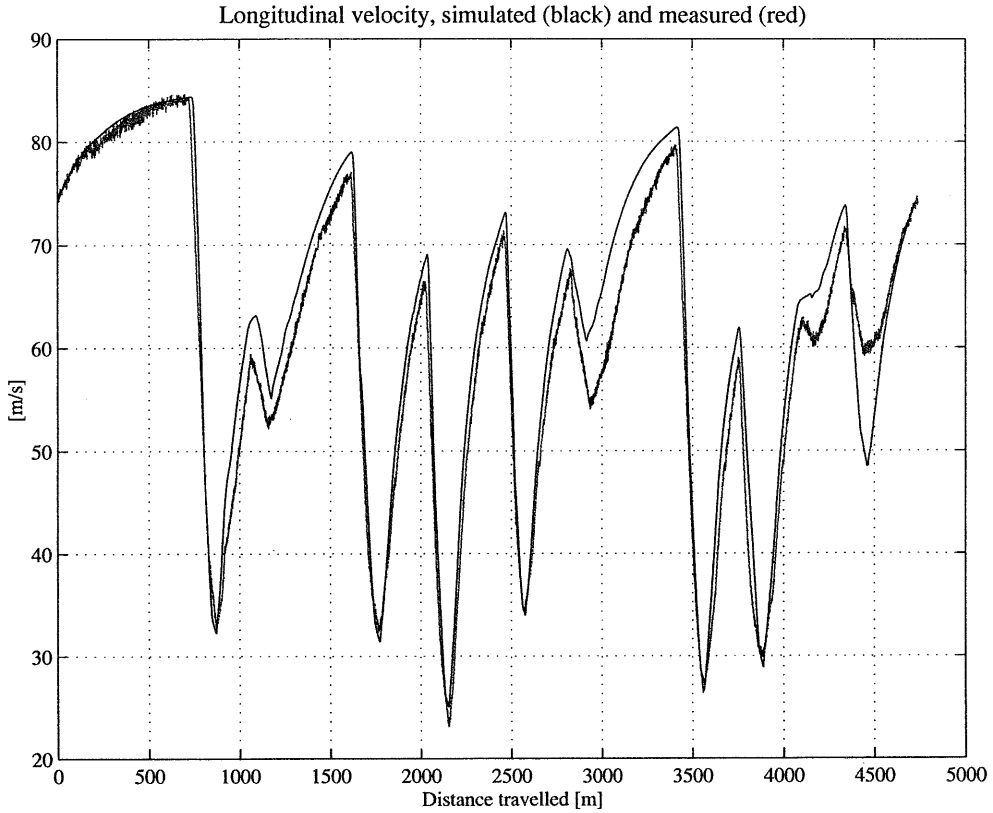


Figure 9.7 Computed and measured vehicle longitudinal velocity, Barcelona.

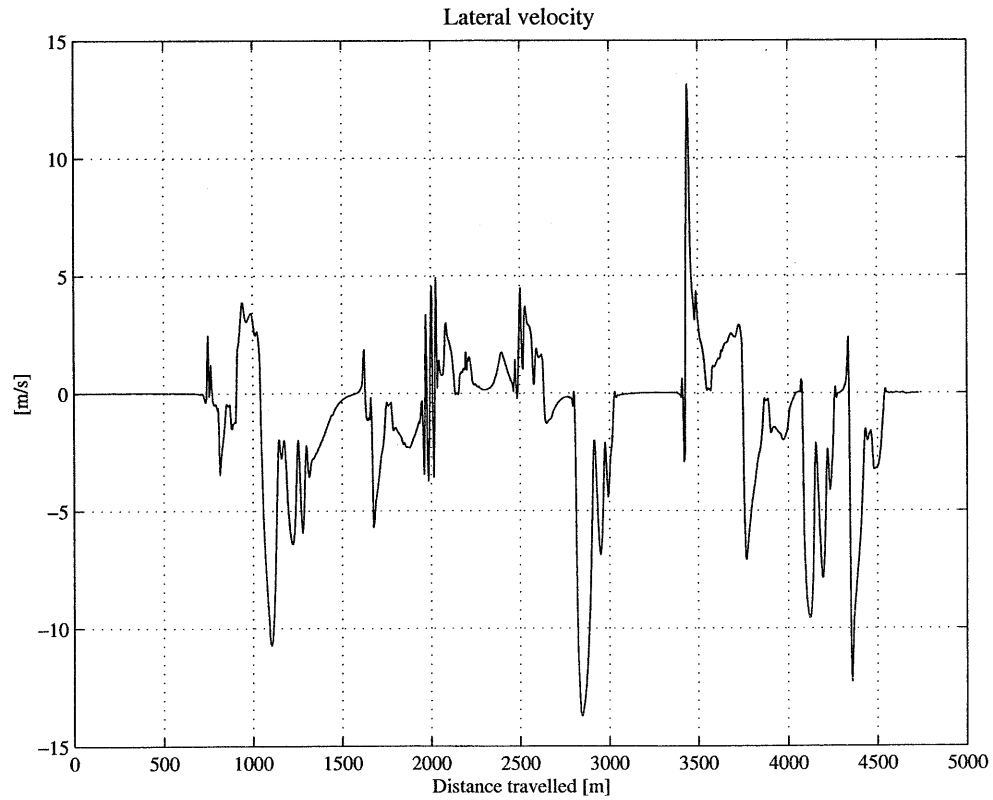


Figure 9.8 Vehicle lateral velocity, Barcelona.

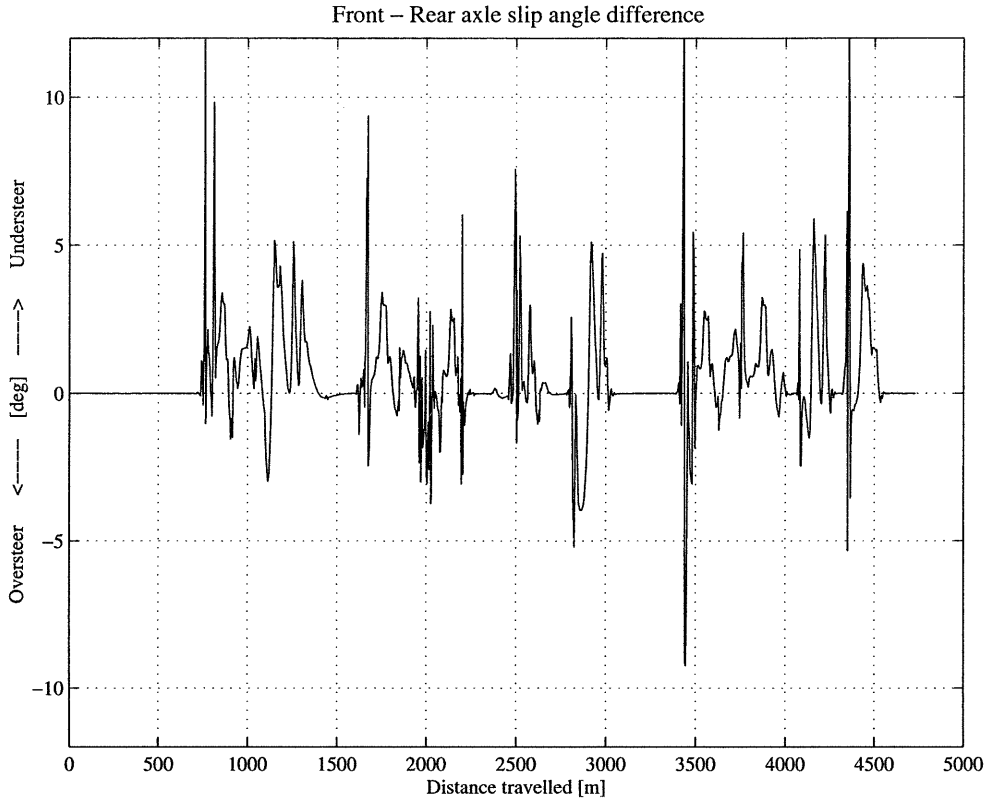


Figure 9.9 Front and rear slip angle difference, Barcelona.

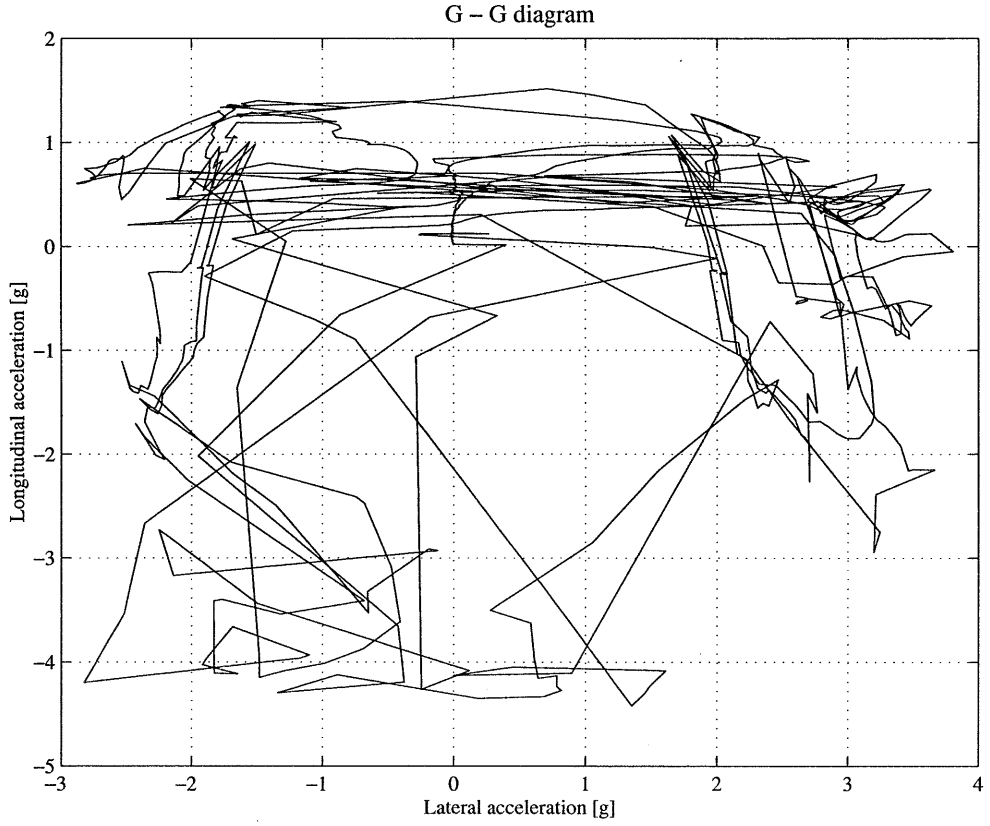


Figure 9.10 Vehicle centre of mass g-g diagram, Barcelona.

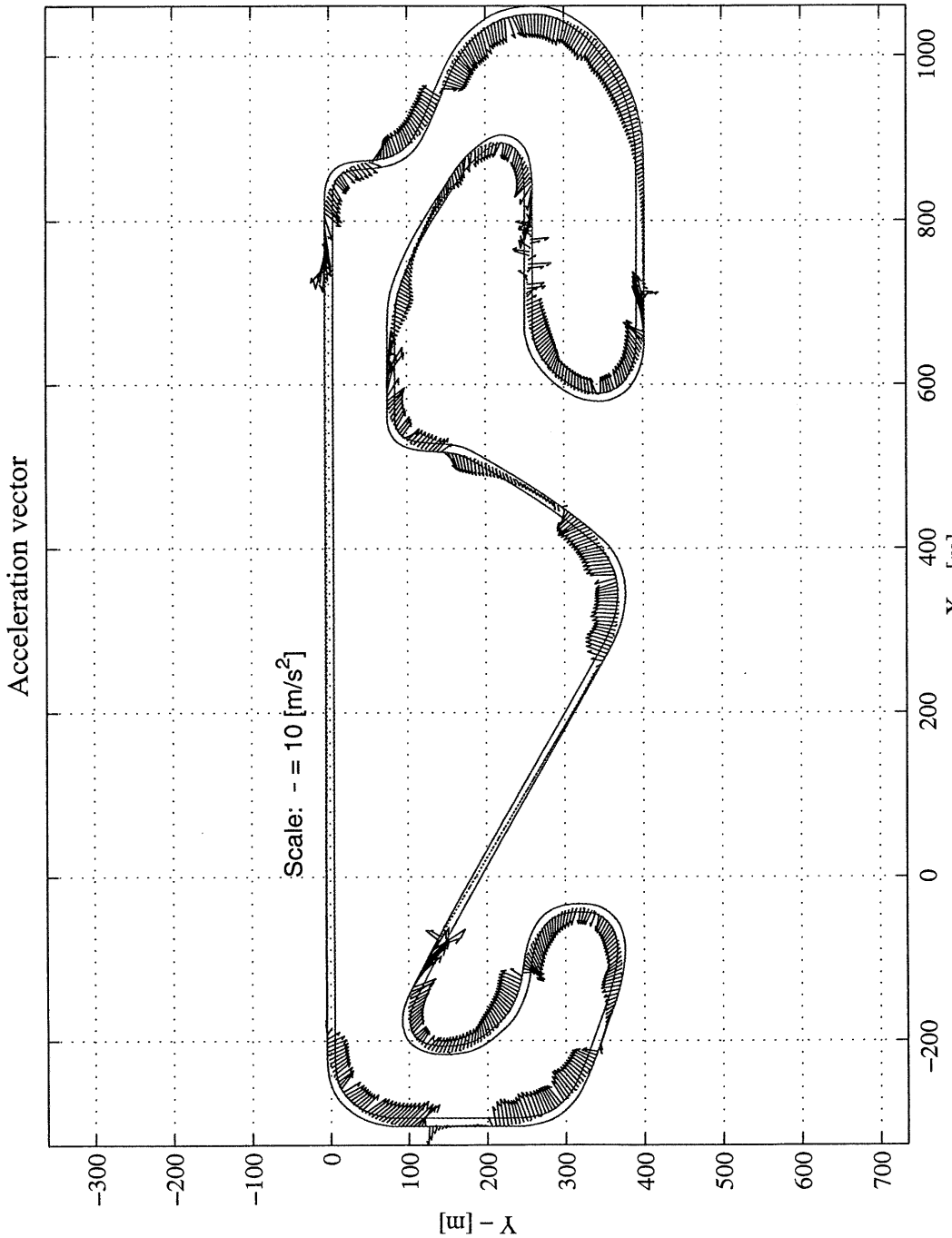


Figure 9.11 Acceleration vector of the vehicle centre of mass, Barcelona.

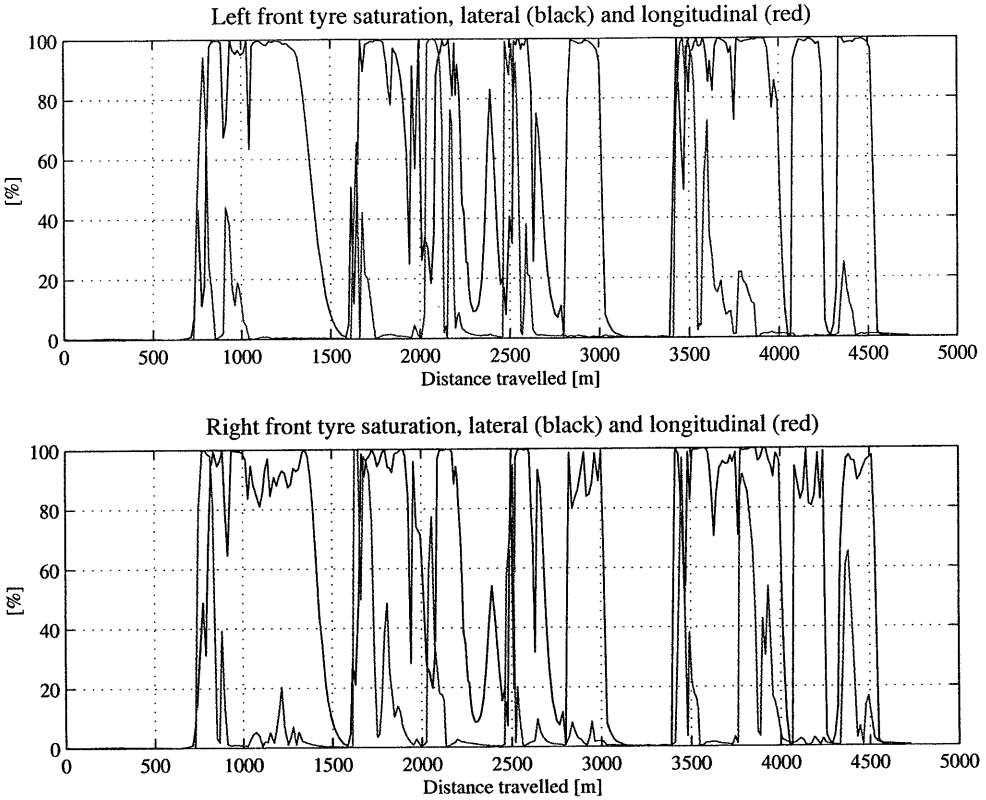


Figure 9.12 Front wheels tyre utilisation indexes, Barcelona.

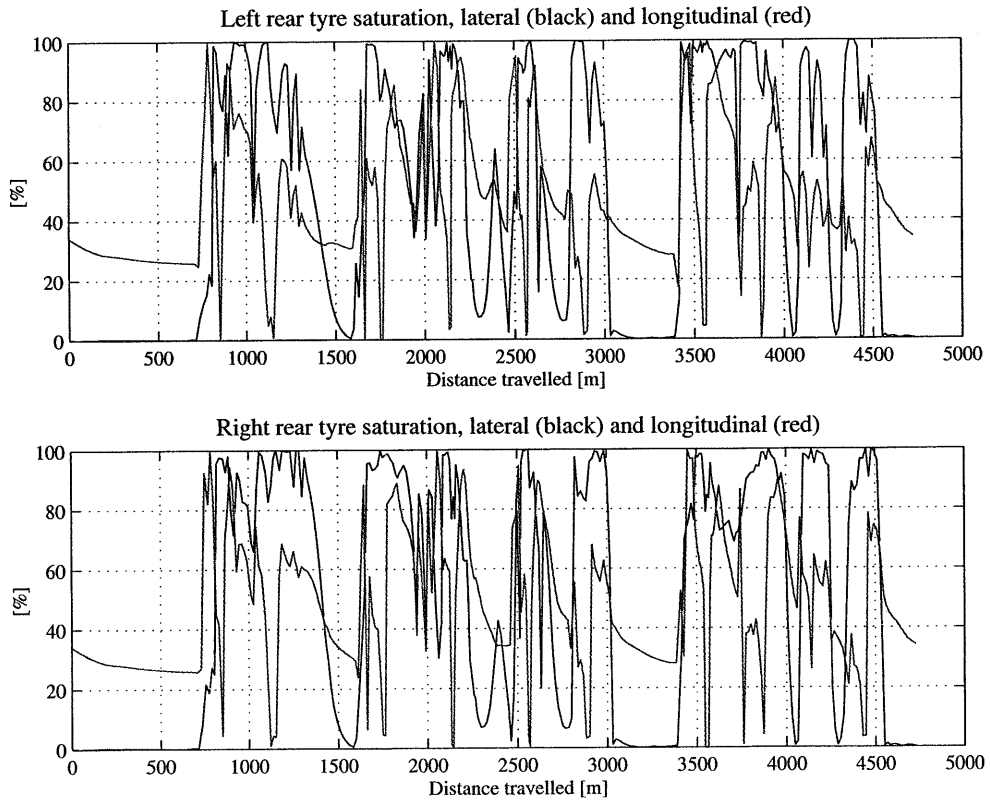


Figure 9.13 Rear wheels tyre utilisation indexes, Barcelona.

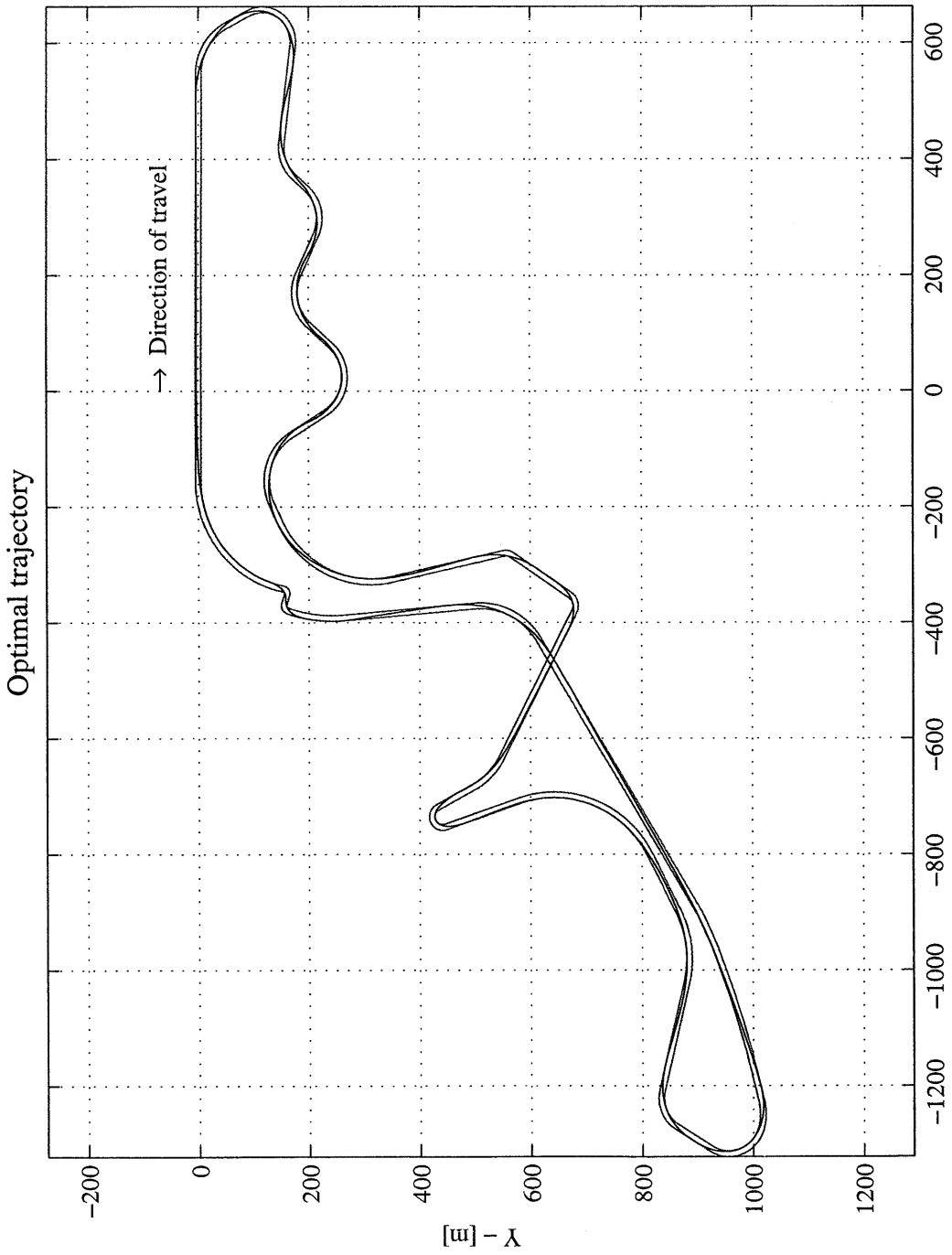


Figure 9.14 Optimal vehicle trajectory, Suzuka circuit.

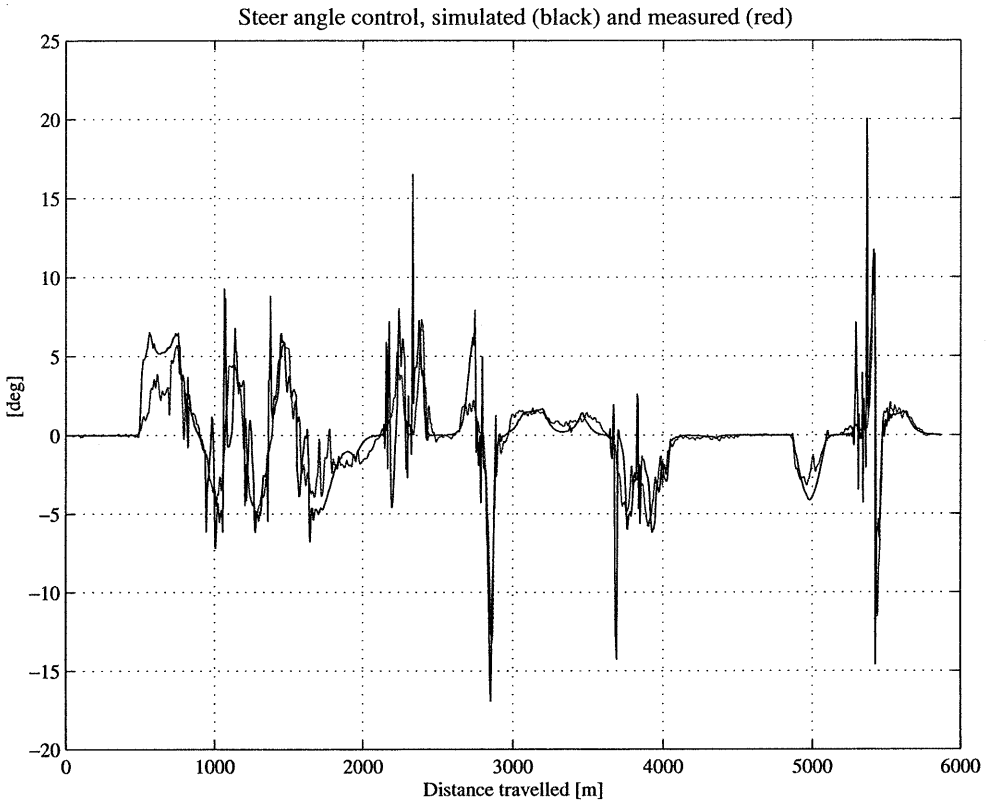


Figure 9.15 Computed and measured steer angle, Suzuka.

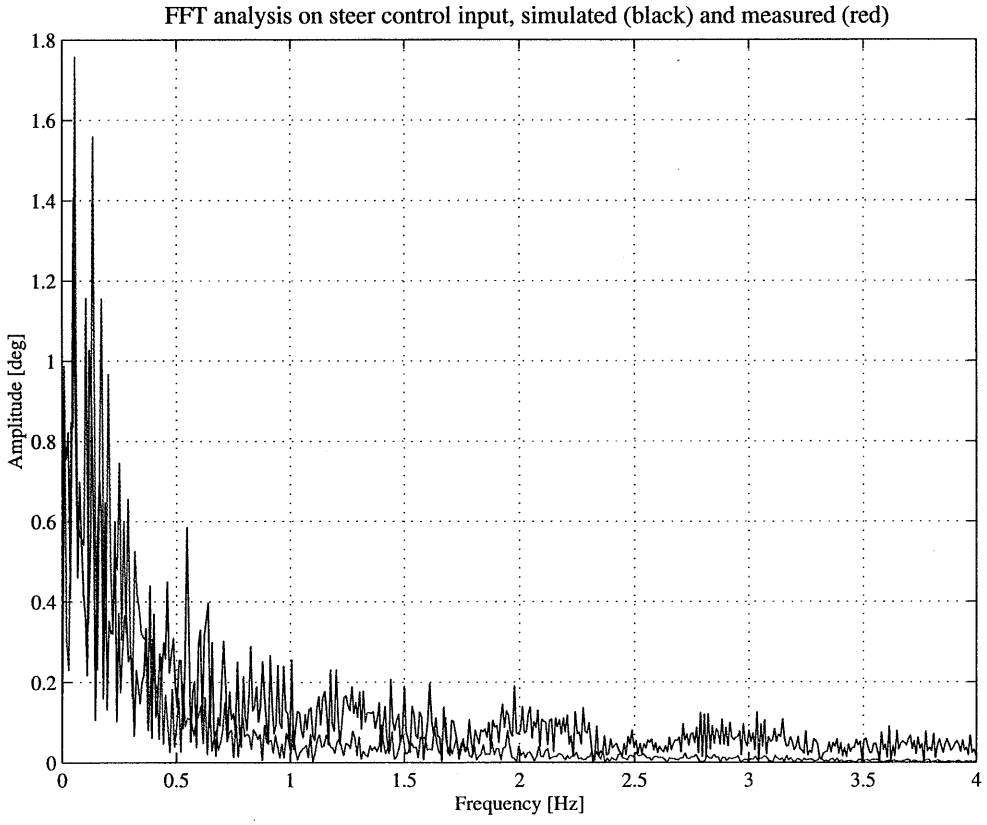


Figure 9.16 FFT analysis of computed and measured steer angle, Suzuka.

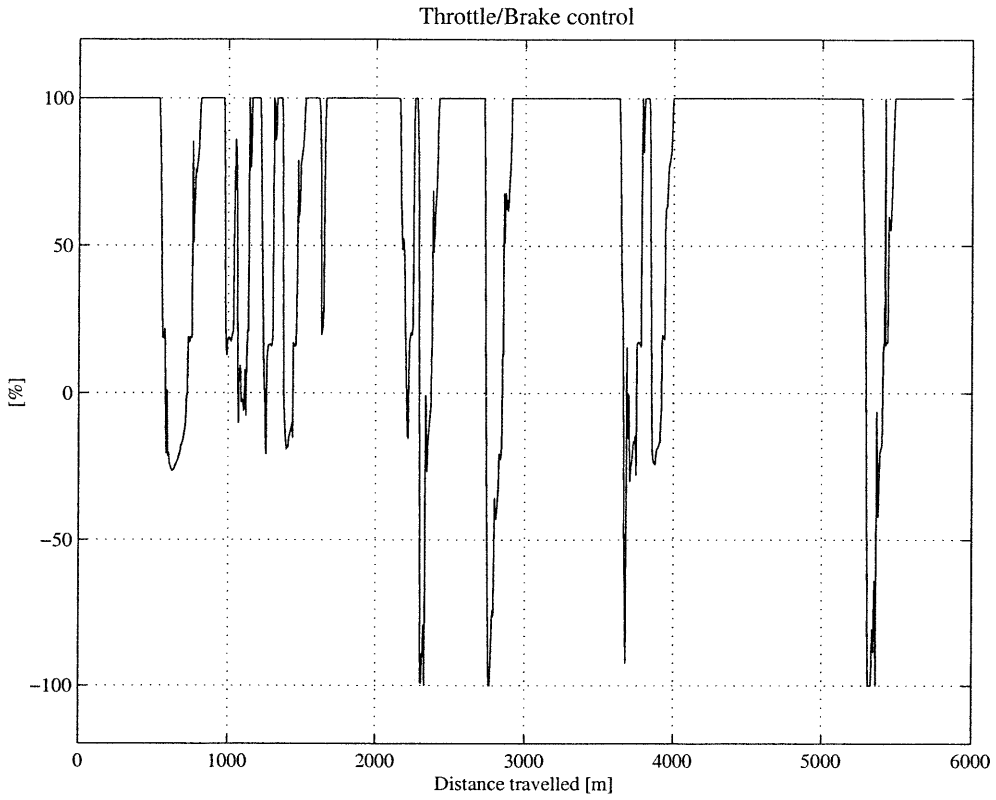


Figure 9.17 Throttle and brake control input, Suzuka.

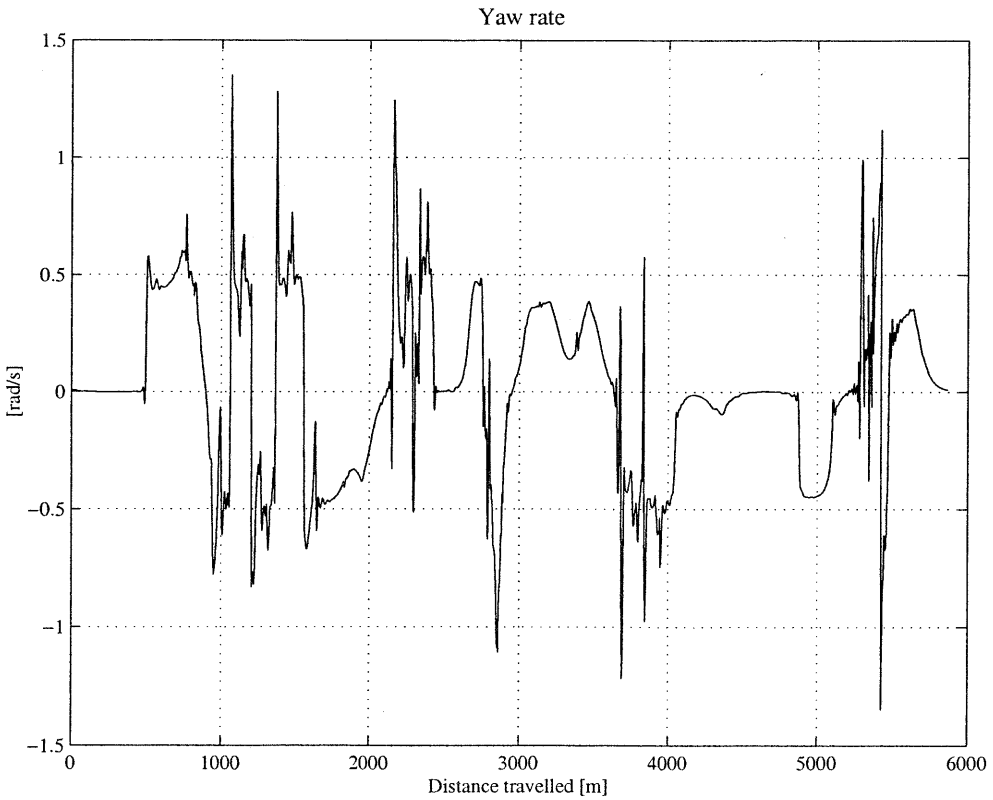


Figure 9.18 Vehicle yaw rate.

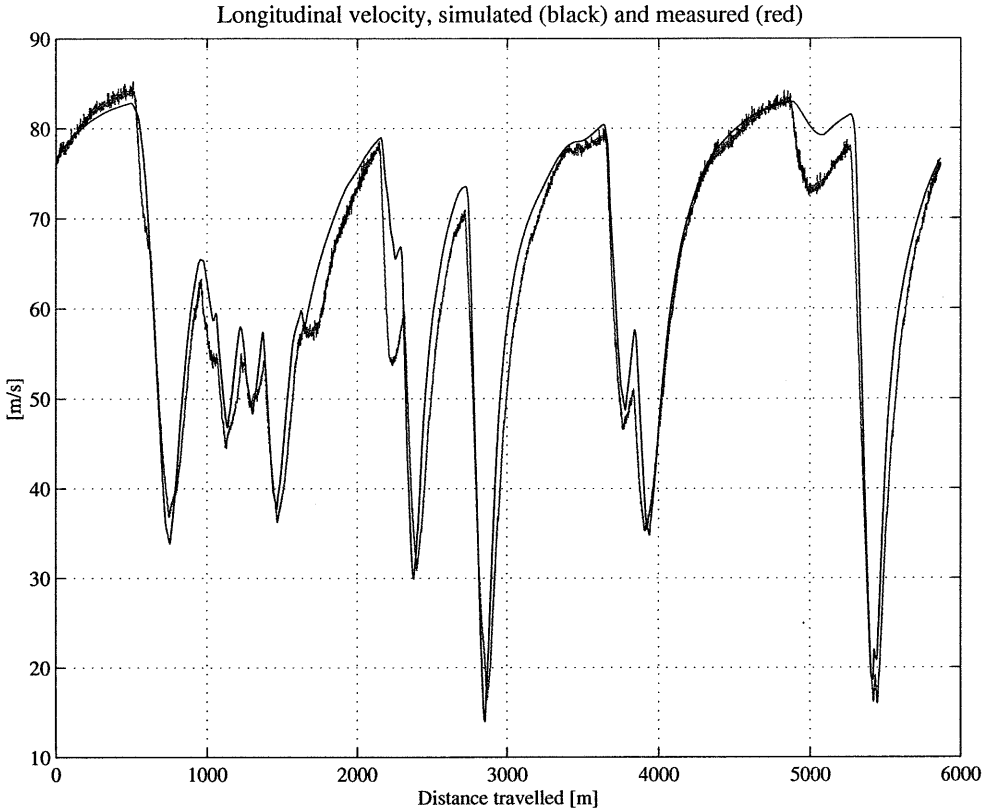


Figure 9.19 Computed vehicle longitudinal velocity compared with measured velocity.

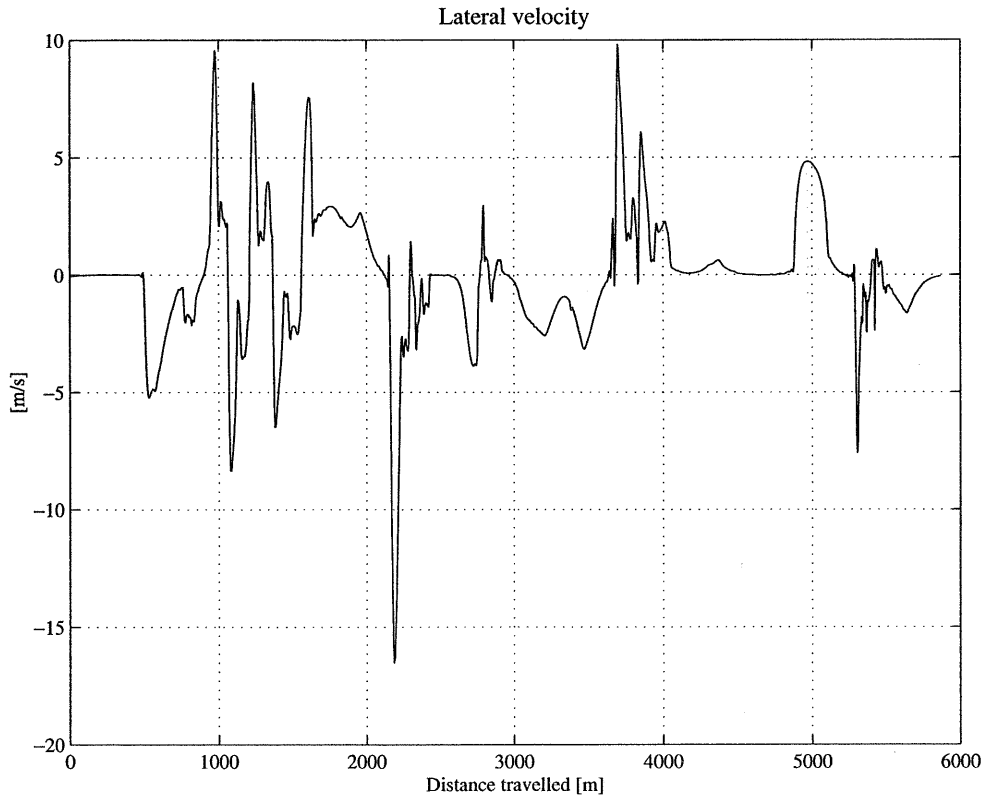


Figure 9.20 Vehicle lateral velocity, Suzuka.

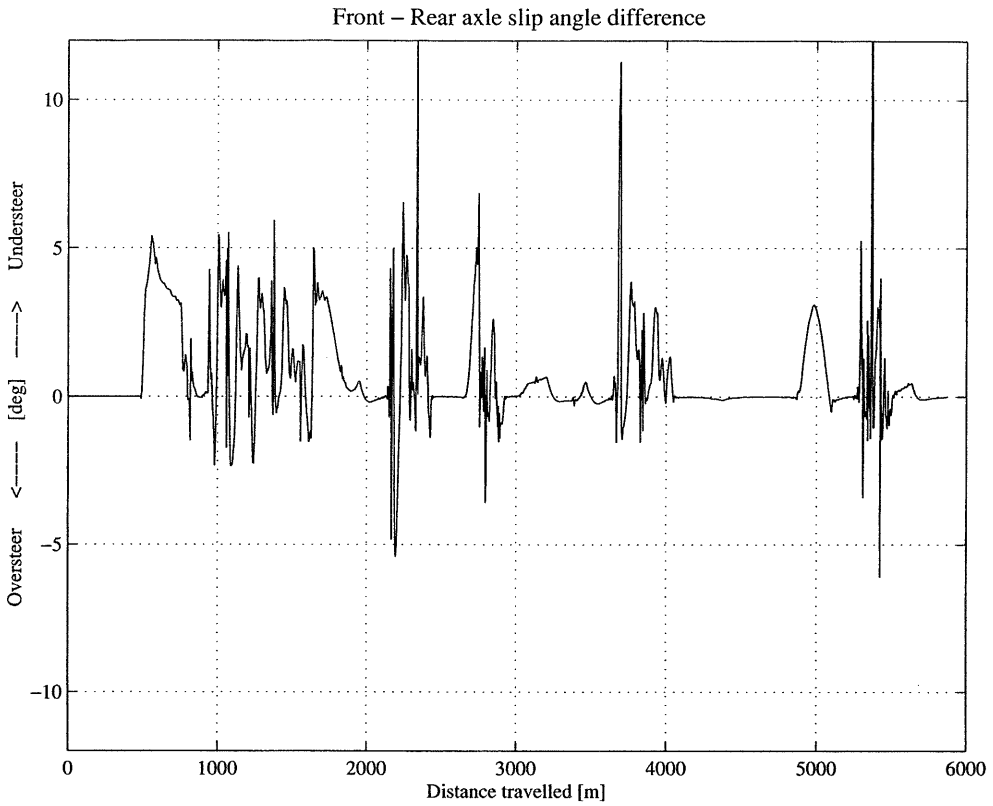


Figure 9.21 Front and rear slip angle difference, Suzuka.

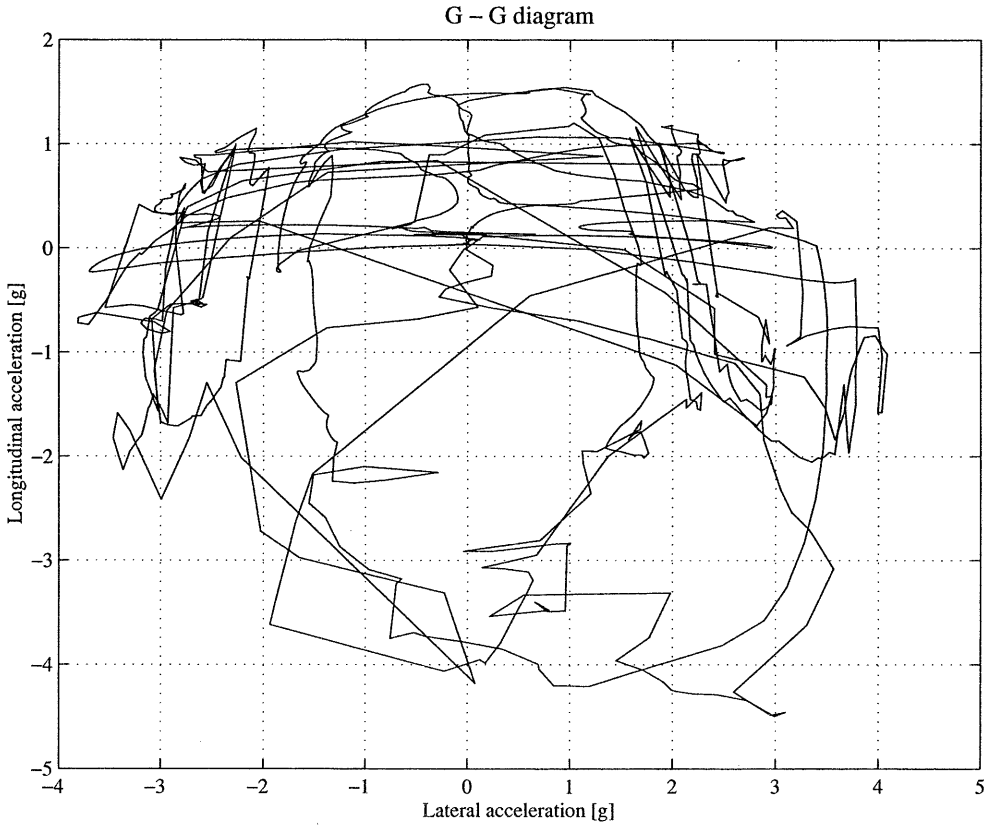


Figure 9.22 Vehicle centre of mass g-g diagram, Suzuka.

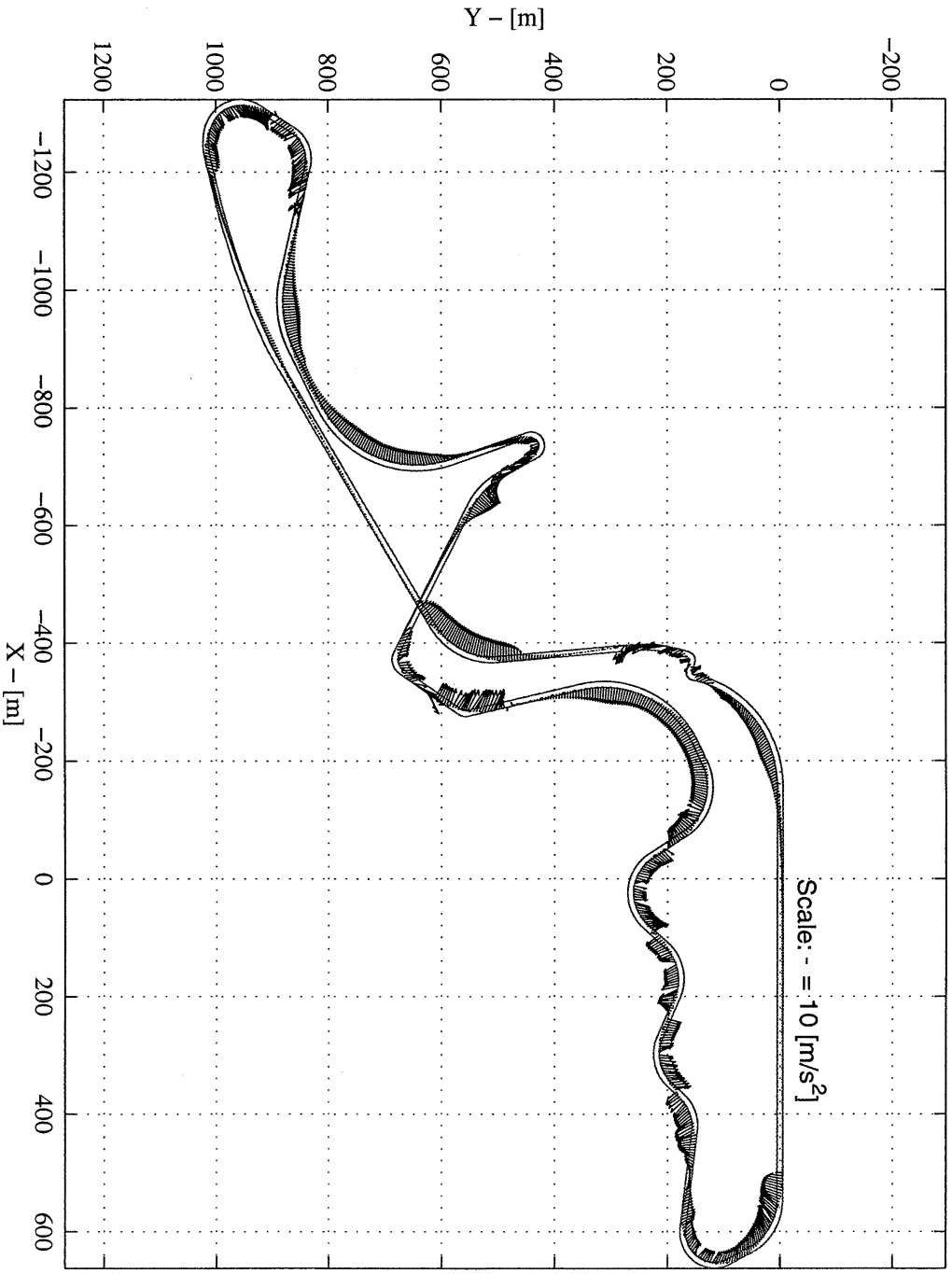


Figure 9.23 Acceleration vector of the vehicle centre of mass, Suzuka.

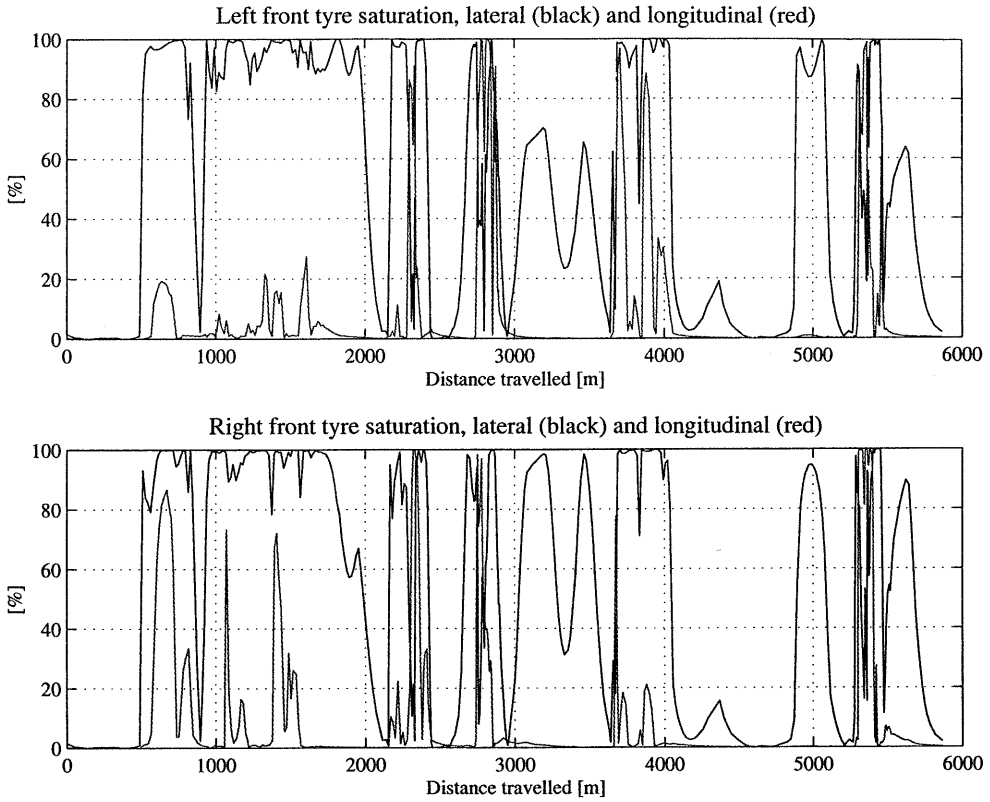


Figure 9.24 Front wheels tyre utilisation indexes, Suzuka.

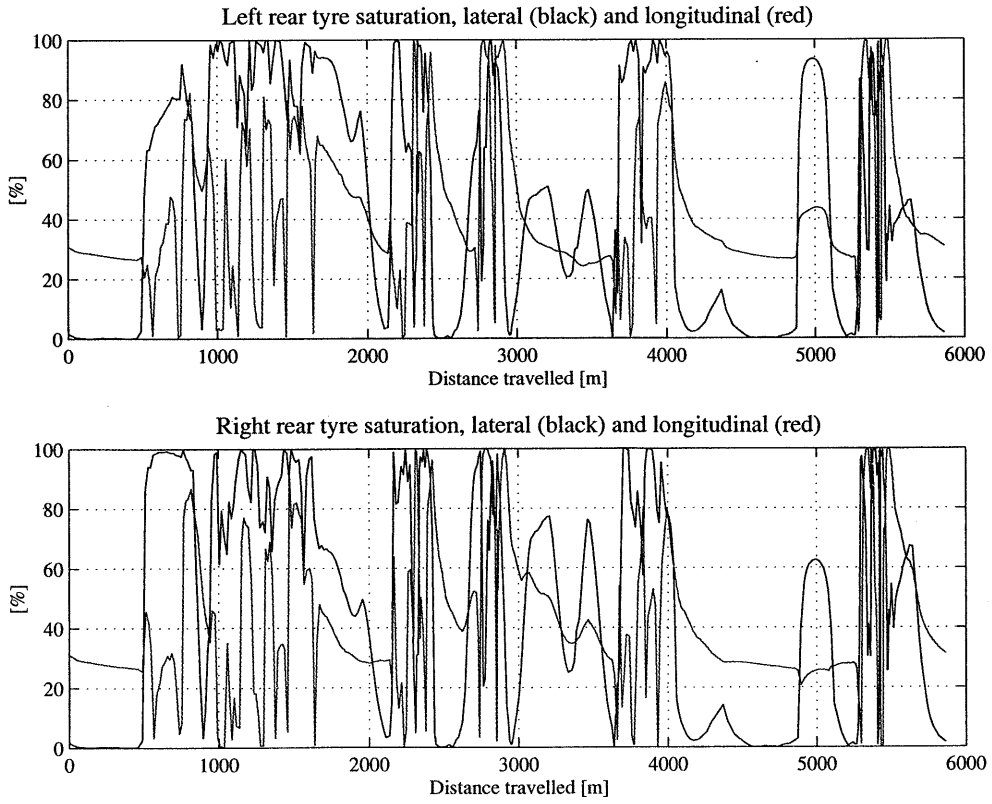


Figure 9.25 Rear wheels tyre utilisation indexes, Suzuka.

9.4. FORMULA ONE CAR VS. RALLY CAR

The comparison of the limit behaviour of a Formula One car with a rally car provides an interesting case to further validate the results returned by FastLap. For this purpose the same seven degrees of freedom vehicle model has been employed using a set of parameters which are representative of a rally saloon car on low friction surfaces. This essentially involves much greater yaw moment of inertia, shorter wheel base, higher centre of mass, virtually no aerodynamic downforce and maximum engine power of only 300 HP in comparison with the 750 HP of the Formula One car. Furthermore, the tyre model has been re-scaled so that the maximum friction coefficient is limited to about 0.6 with the tyre forces saturating at very large slip quantities, i.e. 25 to 30 degrees slip angle and 25 % longitudinal slip. The tyre characteristics are shown in Appendix A. The only thing which could not be changed without modifying the vehicle model was the drive train. Our rally car is rear wheel drive only.

On a low-friction surface the vehicle yaw mode becomes under-damped and oscillatory. Professional rally drivers take advantage of the vehicle natural dynamics by sliding the car in order to enhance the vehicle yaw rate when turning, ultimately achieving greater cornering speeds. It is typical for a rally driver approaching a sharp turn after a straight to apply an oscillatory steer control input much in advance from the corner, with the purpose of exciting the vehicle yaw mode. When the vehicle finally enters the turn with greater yaw rate and speed, the driver must apply a steer control input with a different phase in order to damp the oscillation. A similar strategy applies when changing direction from one turn to the next. With the car already proceeding with large side slip angles, for example on a right hand turn approaching a left hand one, the driver would quickly apply a sharp steer input to the right, which upsets the delicate car equilibrium, and then quickly steer to the left. The car responds with a rapid variation in yaw rate and changes direction very rapidly. The driver must subsequently control these induced oscillations by applying opposite lock.

The idea here is that if the driving strategy described above is indeed the optimal one which maximises the speed of a rally car in low friction conditions, the optimisation program should find it too. In order to investigate this fact, the long version of the double lane change manoeuvre has been employed. It features a long straight before the first turn which allows sufficient space for the vehicle to accelerate, brake and apply steer control input much in advance of the first corner, if needed. The same manoeuvre has been solved using the Formula One general vehicle set-up and the results are compared with those for the rally car. Five runs from different starting solutions have been performed, and Table 9.5 summarises the results.

FORMULA ONE VS. RALLY CAR, LONG DOUBLE LANE CHANGE				
	Formula One car		**Rally car**	
	Manoeuvre time	Num. of iter.	Manoeuvre time	Num. of iter.
Run N. 1	10" 363	108	18" 527	60
Run N. 2	10" 444	86	18" 526	62
Run N. 3	10" 388	96	**18" 526**	72
Run N. 4	**10" 362**	86	18" 527	88
Run N. 5	10" 387	66	18" 526	58

Table 9.5 Formula One car and rally car comparison.

Figures 9.26 and 9.27 show the optimal trajectory for the two cases. Particularly, the thick lines represent the car with its attitude angle along the manoeuvre. Clearly, the rally car proceeds with a much larger side slip angle as a consequence of the tyres generating maximum lateral force at very large slip angles. Figure 9.28 shows the steer angle control for both cars. The control input for the rally car relates very well with the kind of control that a rally driver would apply. Approaching the first right hand turn, steer angle control input is applied starting as early as s equal to 75 [m], firstly to the right, then to the left and finally to the right. The car response is visible in figure 9.26, with the vehicle sliding initially towards the direction opposite to the first right hand turn, at s equal to 80 to 100 [m]. Also the build up of positive lateral velocity[1], see figure 9.32, shows clearly the car attitude. Eventually, the car settles into the right hand turn and the steer control is decreased to nearly zero in order to control the slide, at s equal to 180[m]. At this point the first change of direction occurs, and the steer control input is firstly a very sharp right hand side angle, and then it rapidly shifts to the left. This is expected in a rally car manoeuvre. It allows to take advantage of the car oscillatory behaviour and results in a very rapid change of direction. Next, the car proceeds on the long left hand turn and slowly settles to a large side slip angle, compensated by a progressive reduction of the steer angle, until the steer angle actually becomes opposite to the direction of turning at s equal to 230 [m]. Finally, the last change of direction occurs at s equal to 275 [m], with the steer angle always applied in opposite direction with respect to the direction of turn. Here we do not observe the same kind of steer angle control which characterised the previous change of direction. However, the slide in the final part of the manoeuvre is enhanced anyway by the fact that the vehicle is accelerating, as is seen in figure 9.29, the longitudinal control.

Figure 9.30 shows the vehicle yaw rate. The oscillatory behaviour of the rally car is evident and the peaks of the yaw rate when the vehicle changes direction clearly support the idea that the optimal strategy for driving a rally car on low friction surfaces must take advantage of such behaviour. However, the Formula One car too shows an oscillatory behaviour to some extent, especially before the first right hand turn. This is not inexplicable, even though not usually observed in circuit racing driving. In fact, the combination of vehicle parameters and tyre data for the vehicle model representing a Formula One car which is used here result in both front and rear axles saturating at the same time, as was shown in the previous paragraph. This, too, may cause the vehicle to be oscillatory on the limit and the optimisation algorithm is capable to take advantage of this. This is visible for example from the rather large vehicle lateral velocity for the Formula One car through the first corner, see figure 9.32. Nobody would expect, though, a professional Formula One driver to steer the car the wrong way before a corner!

[1] Positive lateral velocity is towards the right hand side of the vehicle.

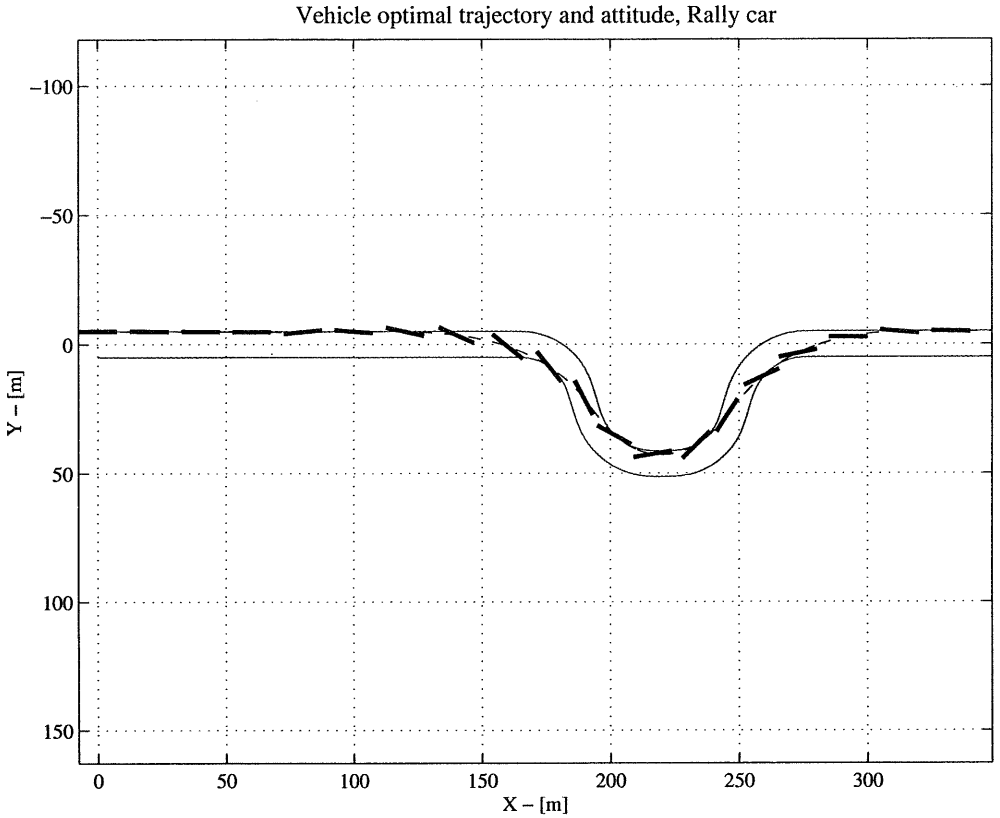


Figure 9.26 Optimal trajectory and attitude for the rally car.

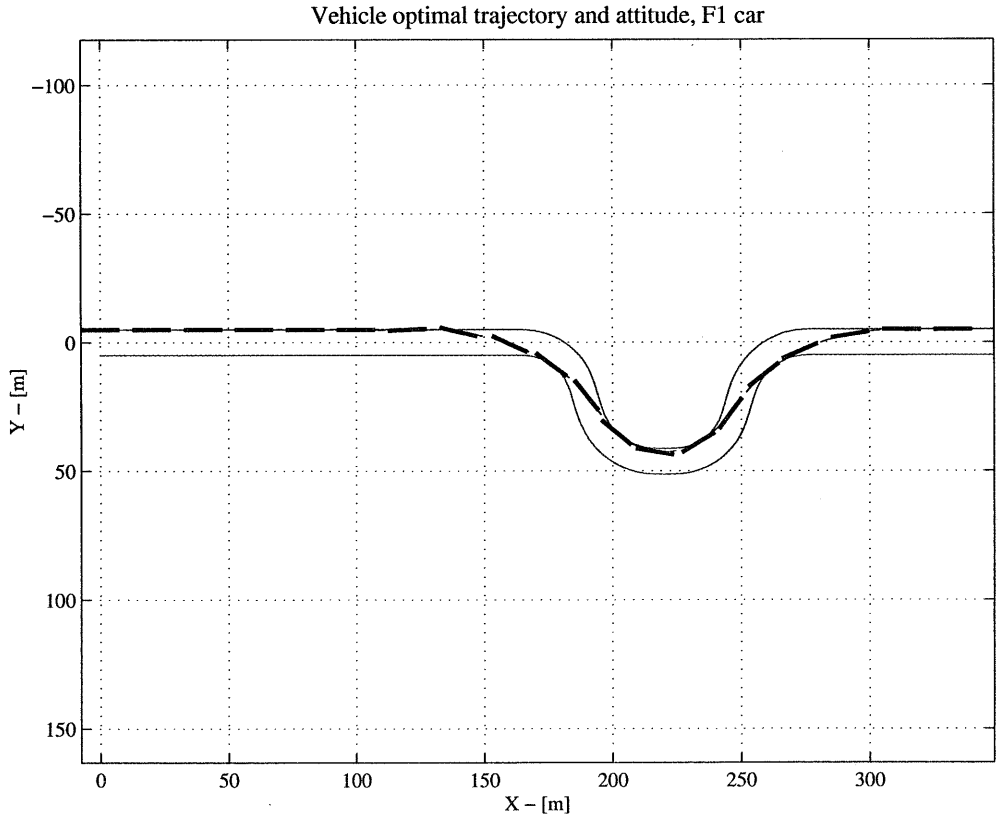


Figure 9.27 Optimal trajectory and attitude for the Formula One car.

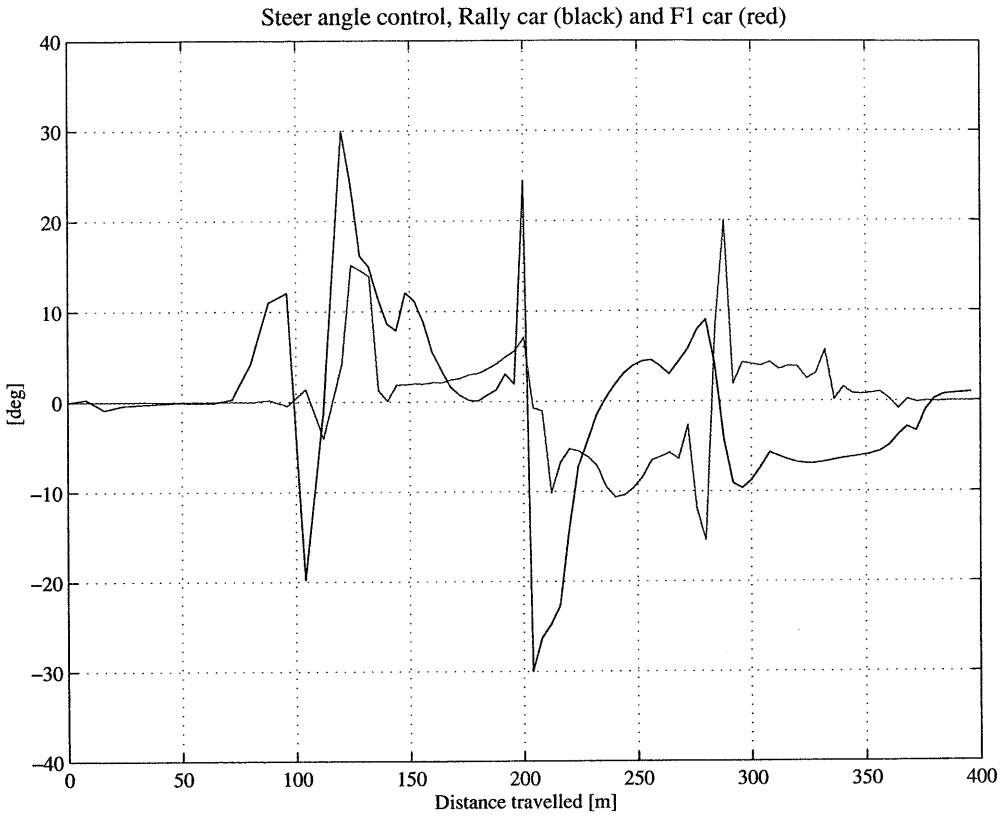


Figure 9.28 Steer angle control input.

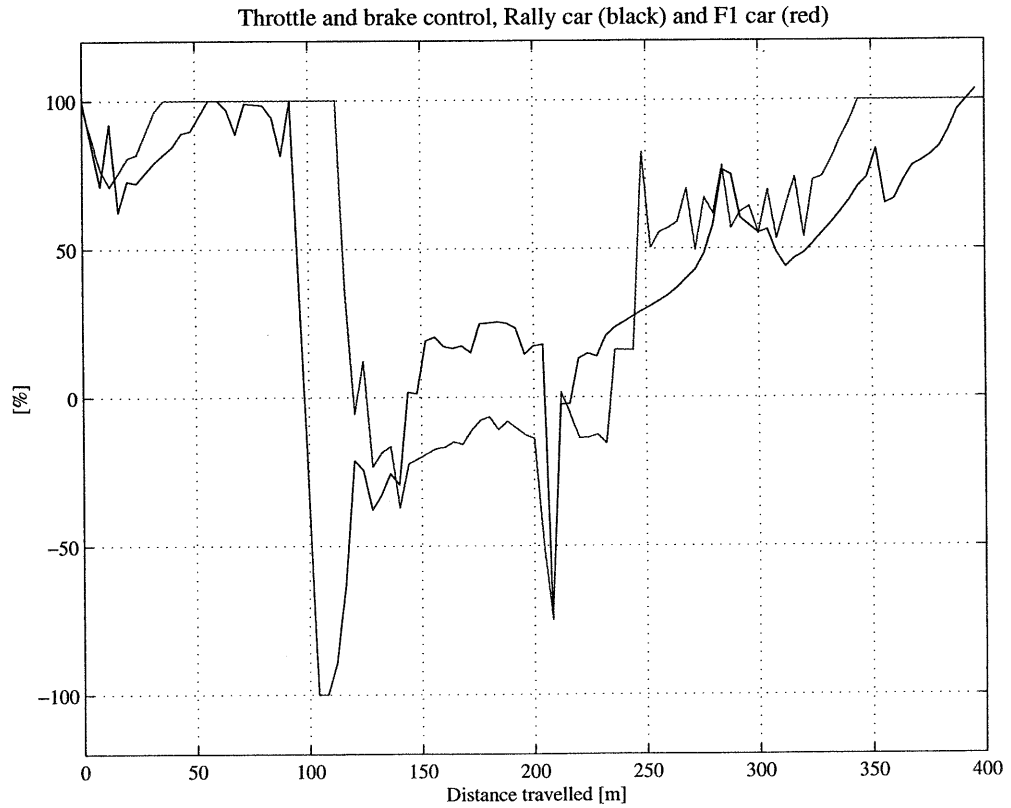


Figure 9.29 Throttle and brake control input.

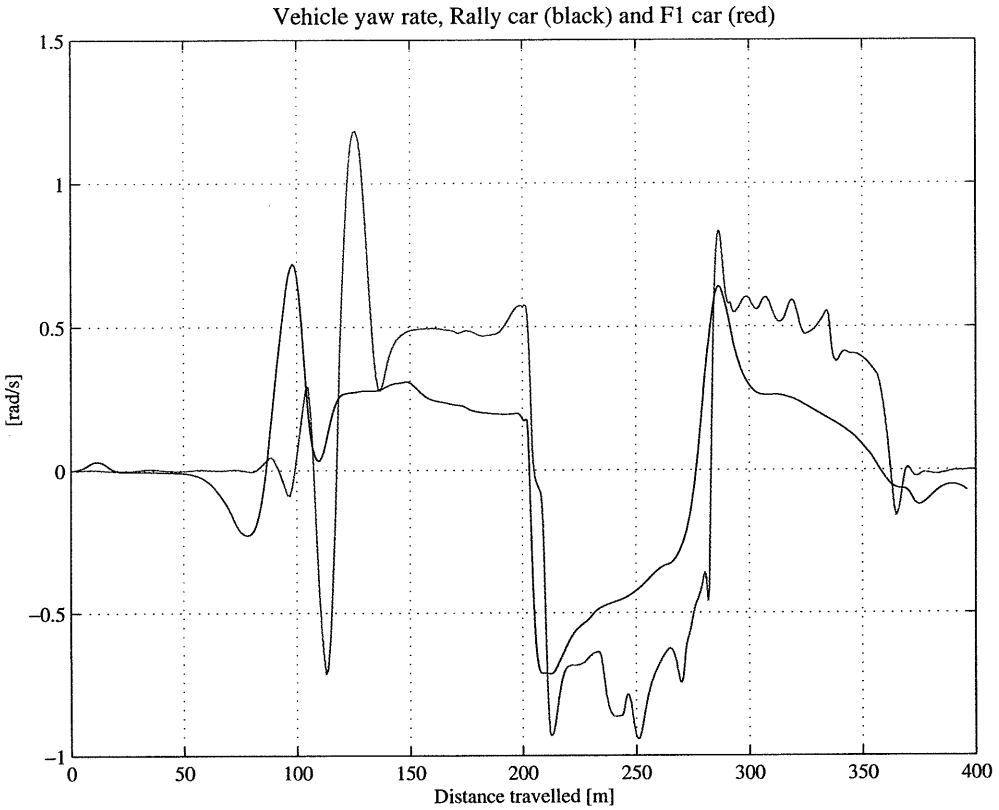


Figure 9.30 Vehicle yaw rate.

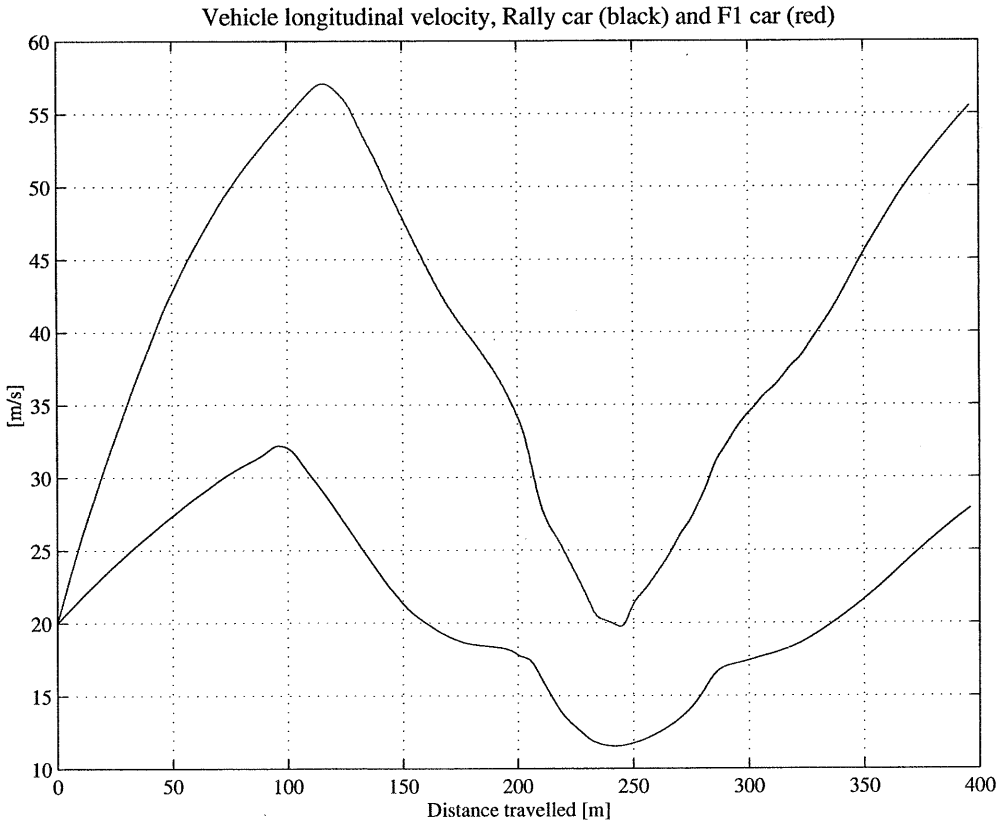


Figure 9.31 Vehicle longitudinal velocity.

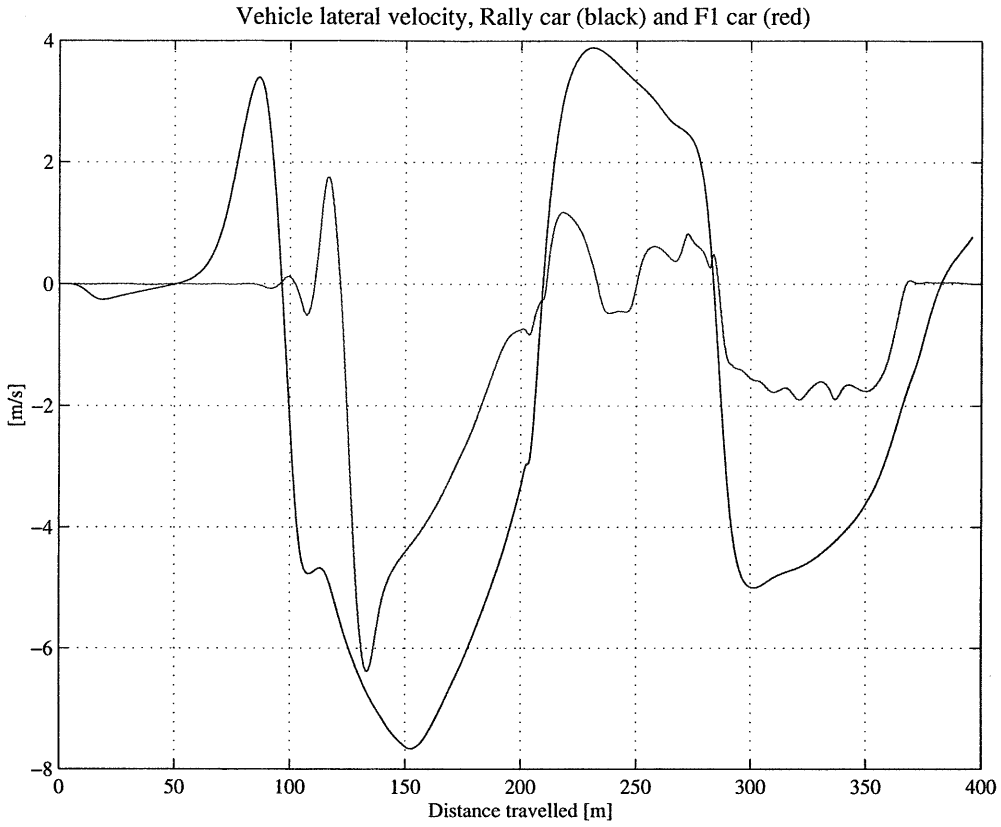


Figure 9.32 Vehicle lateral velocity.

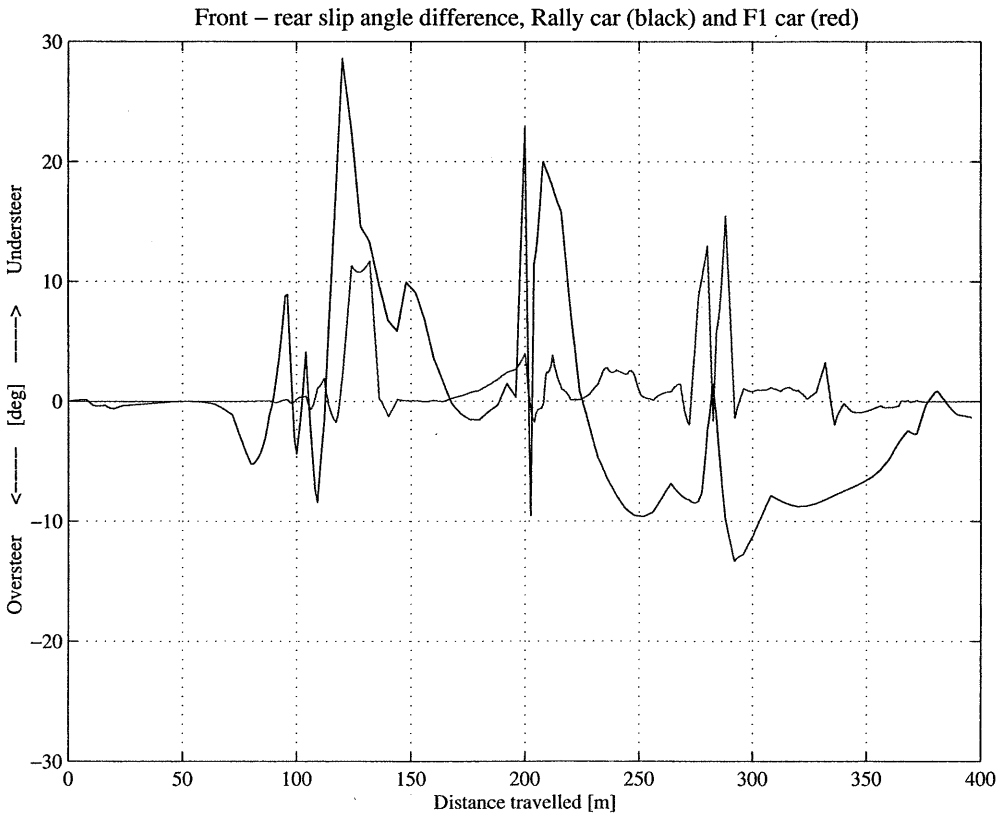


Figure 9.33 Front and rear wheel slip angle difference.

9.5. YAW INERTIA SENSITIVITY ANALYSIS

In this section FastLap is used to investigate the influence of yaw inertia on vehicle performance. For this purpose, the minimum time manoeuvring has been computed several times using the double lane change manoeuvre with increasing values of the vehicle yaw inertia, i.e. from 100 to 10000 [kg·m^2]. For each value of the yaw inertia the optimisation has been repeated four times with different initial trial solutions. Only the default high precision mode for FastLap has been used here.

Figures 9.34 to 9.37 show the optimal solution for the vehicle with default parameters. The reference value of yaw inertia for a Formula 1 car is set to be equal to 700 kg·m^2. Figure 9.34 shows the racing line for the double lane change manoeuvre and figure 9.35 shows the corresponding control histories. The control inputs appear to be rather noisy, as was already observed in §9.3, as a consequence of using linear interpolation for the evaluation of the actual control values. Furthermore, when the vehicle is working on the limits of its tyres, a small variation of the steer angle will not cause a significant variation of the tyre forces. In fact, the tyre lateral forces have a rather flat maximum, as is shown in Appendix A. Figure 9.36 shows the vehicle yaw rate and longitudinal velocity, and figure 9.37 shows the vehicle lateral velocity and the difference between the front and rear slip angles. The large values for the vehicle lateral velocity through the first corners reveals once more the oversteer nature of the car in its default set-up.

The top graph in figure 9.38 shows the minimum manoeuvre time for the various values of yaw inertia. The best and the worst values for the manoeuvre time obtained in the four optimisation runs for each value of yaw inertia are reported. The results are spread in a range of the order of few hundredths of a seconds, apart from the case with the lowest value of yaw inertia where the uncertainty is as large as one tenth of a second. This level of accuracy appears to be sufficient to quantify the sensitivity of the performance of a Formula One race car with respect to yaw inertia, which appears to be not at all significant up to values as large as 2000 [kg·m^2]. Figure 9.38 also shows the average number of iterations required to solve each case corresponding to the various values of yaw inertia. It is interesting to notice that the lower the vehicle yaw inertia, the more iterations are needed, and for the lowest value of inertia convergence was never achieved within the 200 iterations allowed. A real car is expected to become more and more responsive to control inputs at lower values of yaw inertia. In the theoretical world of the lap time optimisation program this translates into greater numerical sensitivity problems.

Figures 9.39 and 9.40 show the front and rear axle lateral forces for four cases, i.e. I_{zz} equal to 500, 1000, 2000 and 5000 [kg·m^2]. As expected, the results show an increasing time lead of the front axle lateral force over the rear axle lateral force for higher yaw inertias. A further analysis has been performed by evaluating the unbalanced yaw moment M_{zz}, i.e. the moment generated by the tyre forces about the yaw axis in order to vary the yaw rate, as follows:

$$M_{zz} = (F_{x1} - F_{x2}) \cdot \frac{t_f}{2} + (F_{x3} - F_{x4}) \cdot \frac{t_r}{2} + (F_{y1} + F_{y2}) \cdot l_f - (F_{y3} + F_{y4}) \cdot l_r \quad \text{Eq. 9.1}$$

The results are shown in figure 9.41 for three values of yaw inertia. A comparison between figure 9.41 and the first of figure 9.36, shows how the unbalanced yaw moment

M_{zz} assumes peak values during the transient phases of the manoeuvre where the yaw rate varies significantly. Moreover, the greater the yaw inertia, the higher are the peaks. However, the length of each transient phase appears to be very short and, in the end, the effect on the minimum manoeuvre time is negligible unless the yaw inertia is increased up to unrealistic values.

Finally, in figures 9.42 to 9.45 the vehicle performance through the double lane change manoeuvre has been compared with the maximum steady-state performance using the g-g diagram (the analytical evaluation of the g-g diagram is described in Appendix C). Figures 9.42 and 9.43 refer to the vehicle with default parameters, while figures 9.44 and 9.45 refer to the vehicle with the greatest yaw inertia, i.e. 10000 [kg·m^2]. The longitudinal and lateral accelerations of the vehicle during the double lane change manoeuvre have been estimated from the state trajectories by using numerical differentiation. The values obtained have been grouped together depending on the vehicle velocity. For example, all the couples of longitudinal and lateral acceleration values that the vehicle is subjected to, when it is travelling at a velocity between 17.5 to 22.5 m/s, have been plotted on the top left graphs of figures 9.42 and 9.44. For the case of the vehicle with the default yaw inertia, most of the points are gathered on the edge of the g-g diagram, showing that most of the optimal manoeuvre is near steady-state conditions. For the case of the greatest yaw inertia, several points lie inside the g-g diagram when the velocity is equal to 25, 30, 35 and 45 [m/s]. In this case the vehicle spends a much longer time in transient phases where the lateral tyre forces act in opposition and this determines the manoeuvre time penalty.

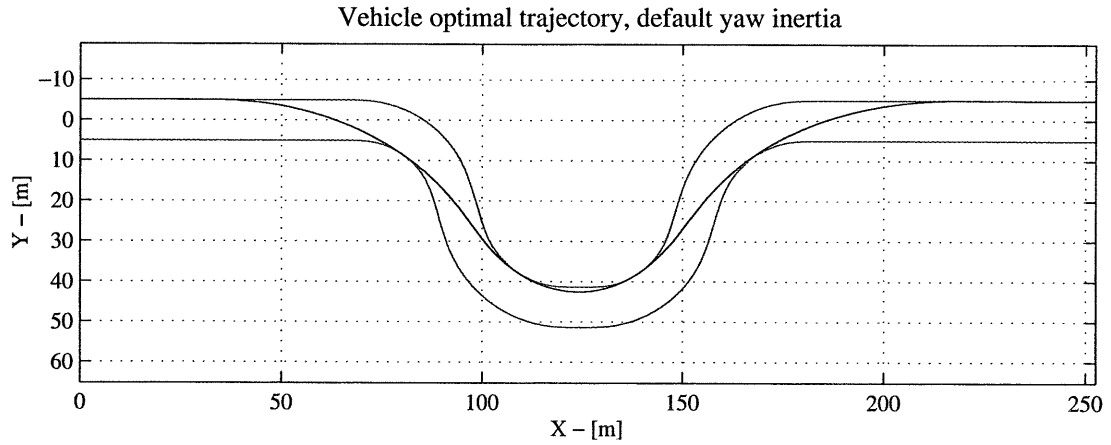


Figure 9.34 Vehicle optimal trajectory, default yaw inertia.

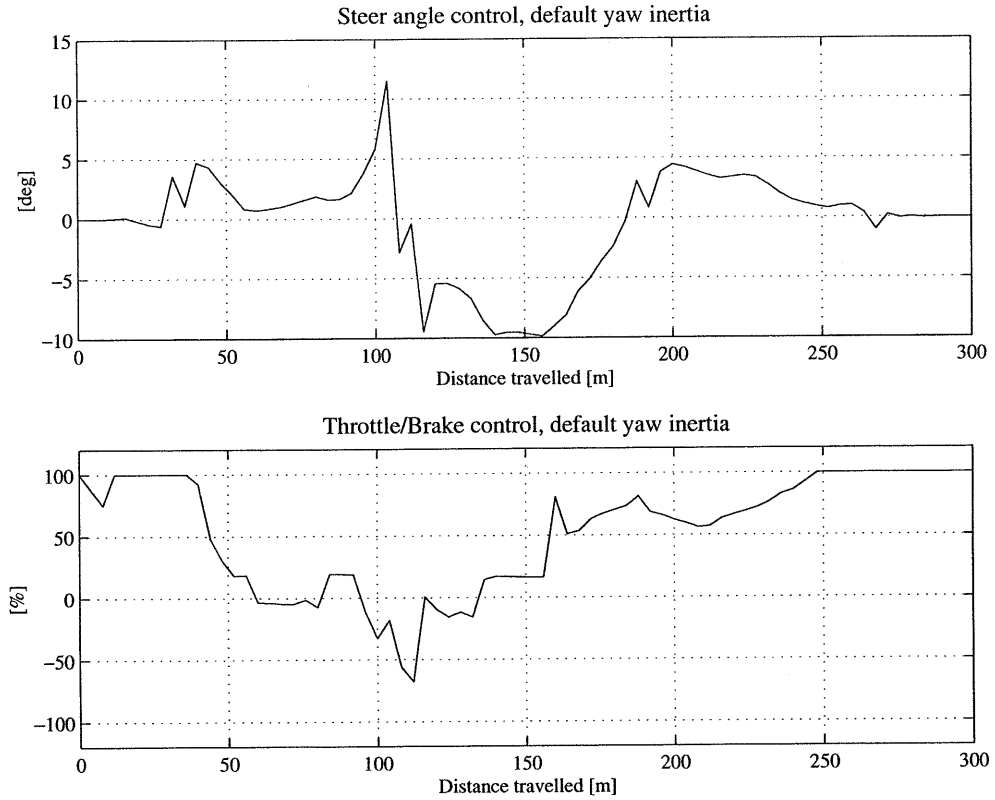


Figure 9.35 Vehicle optimal controls, default yaw inertia.

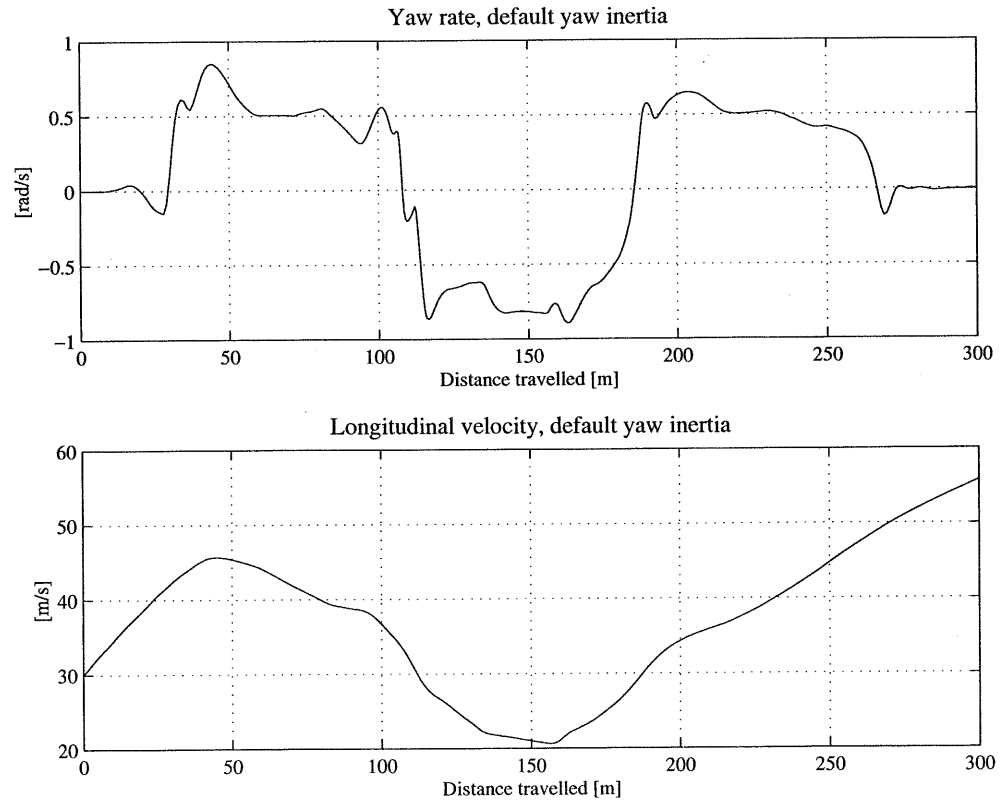


Figure 9.36 Vehicle yaw rate and longitudinal velocity, default yaw inertia.

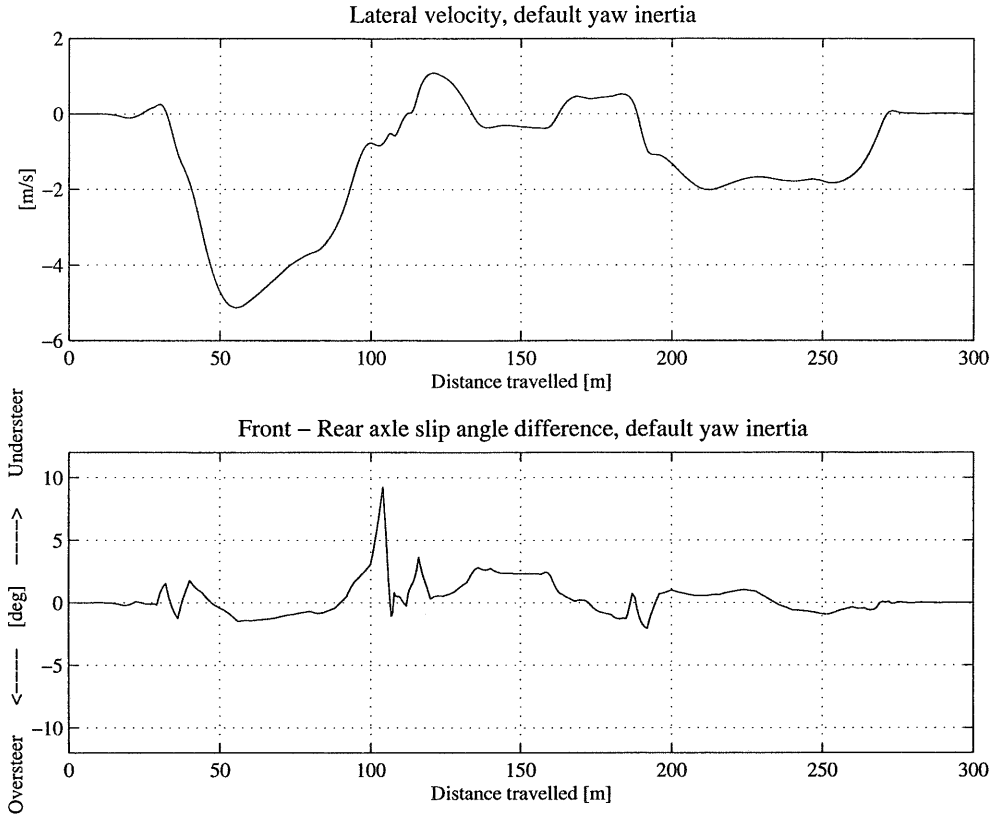


Figure 9.37 Vehicle lateral velocity and slip angle difference, default yaw inertia.

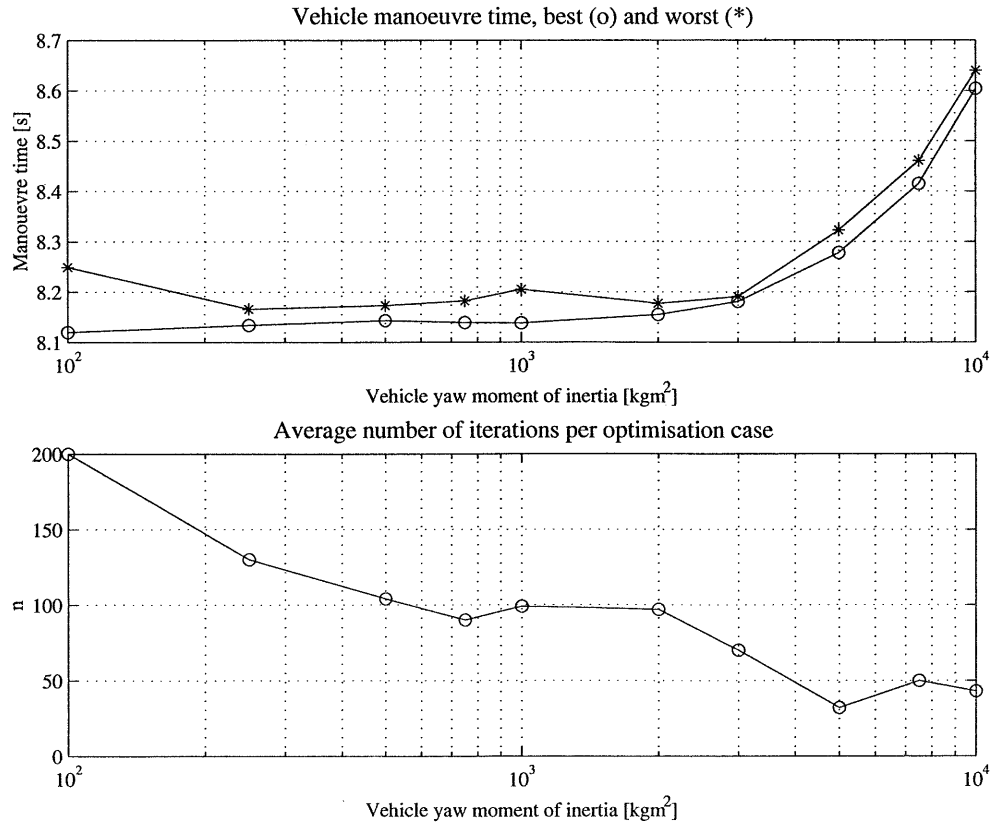


Figure 9.38 Vehicle manoeuvre time and average number of iterations per case.

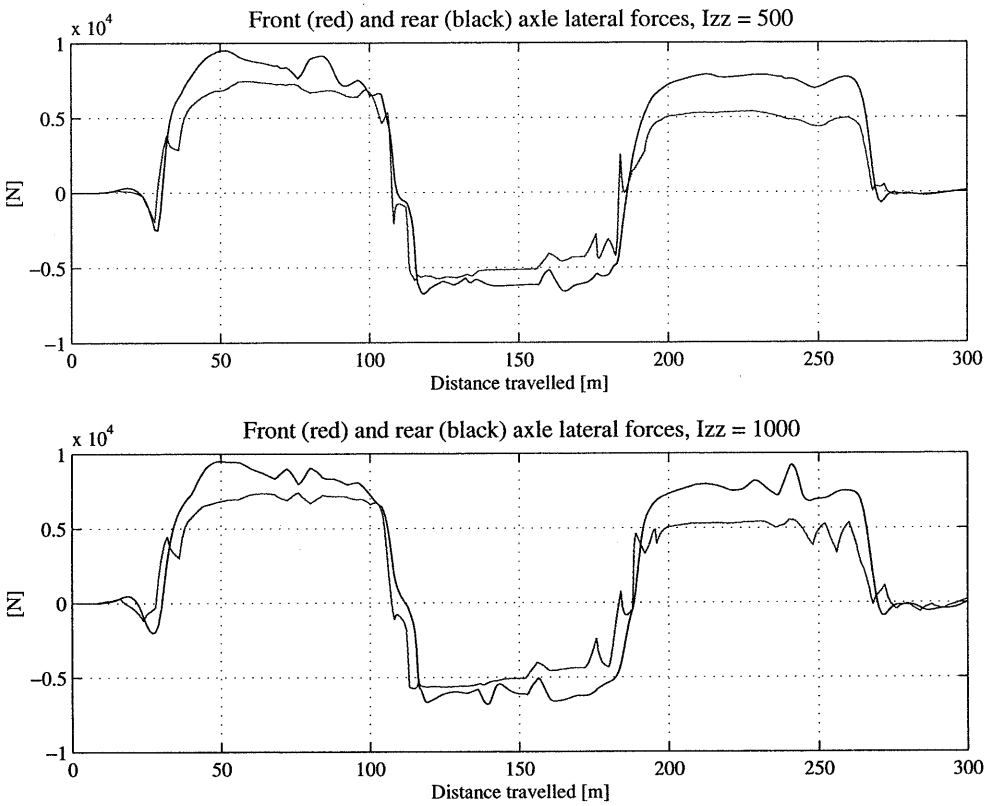


Figure 9.39 Front and rear axle lateral forces, Izz = 500 and 1000 [kgm^2].

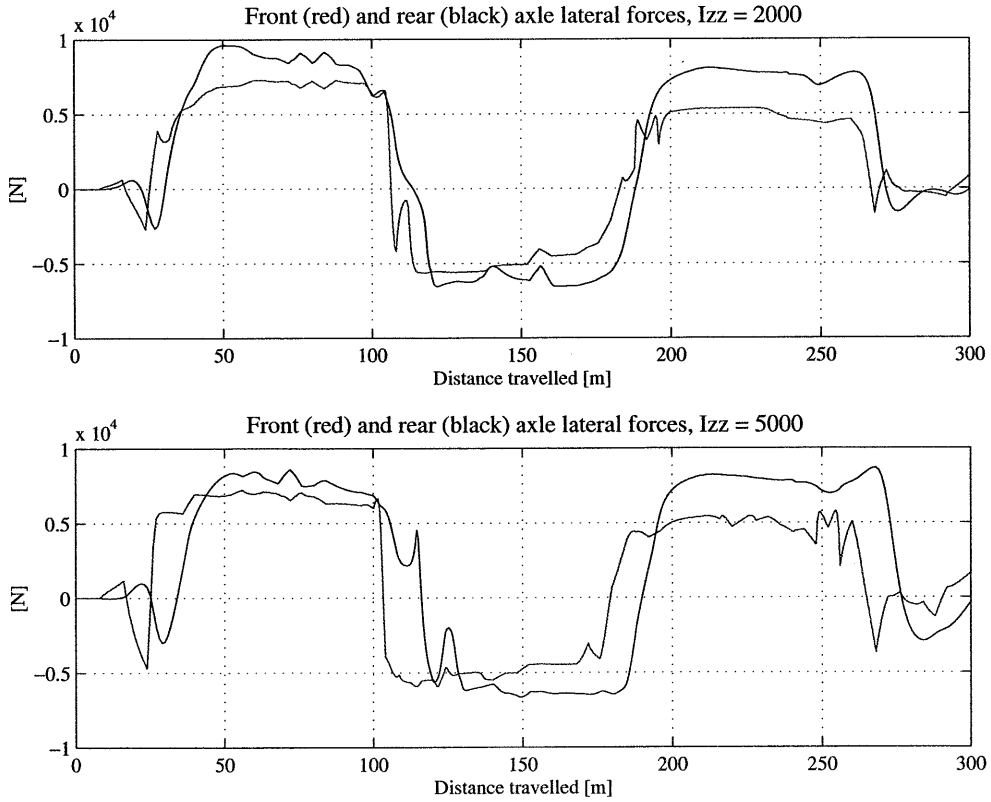


Figure 9.40 Front and rear axle lateral forces, Izz = 2000 and 5000 [kgm^2].

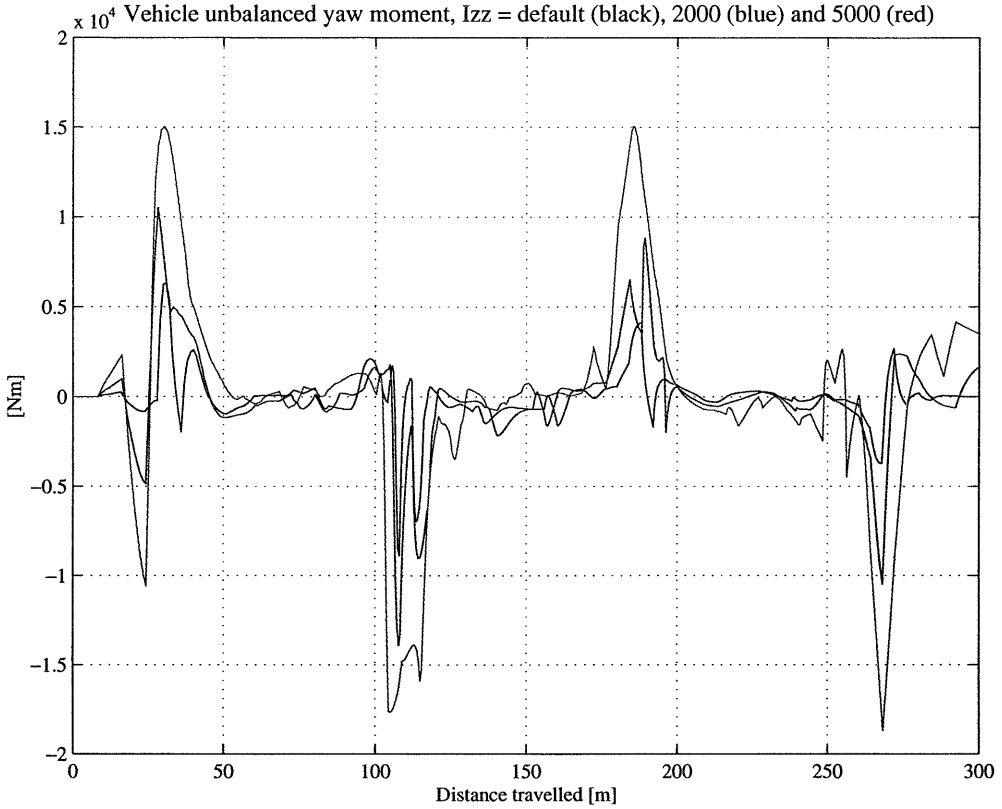


Figure 9.41 Unbalanced yaw moment.

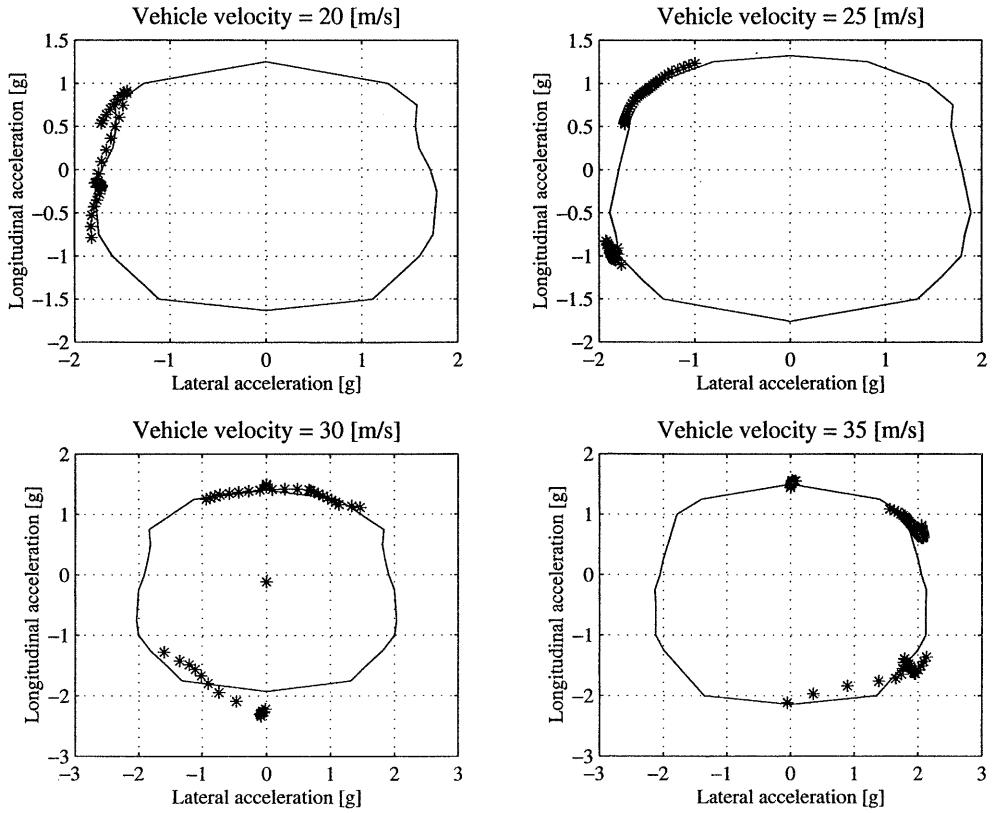


Figure 9.42 g-g analysis of the results, default yaw inertia.

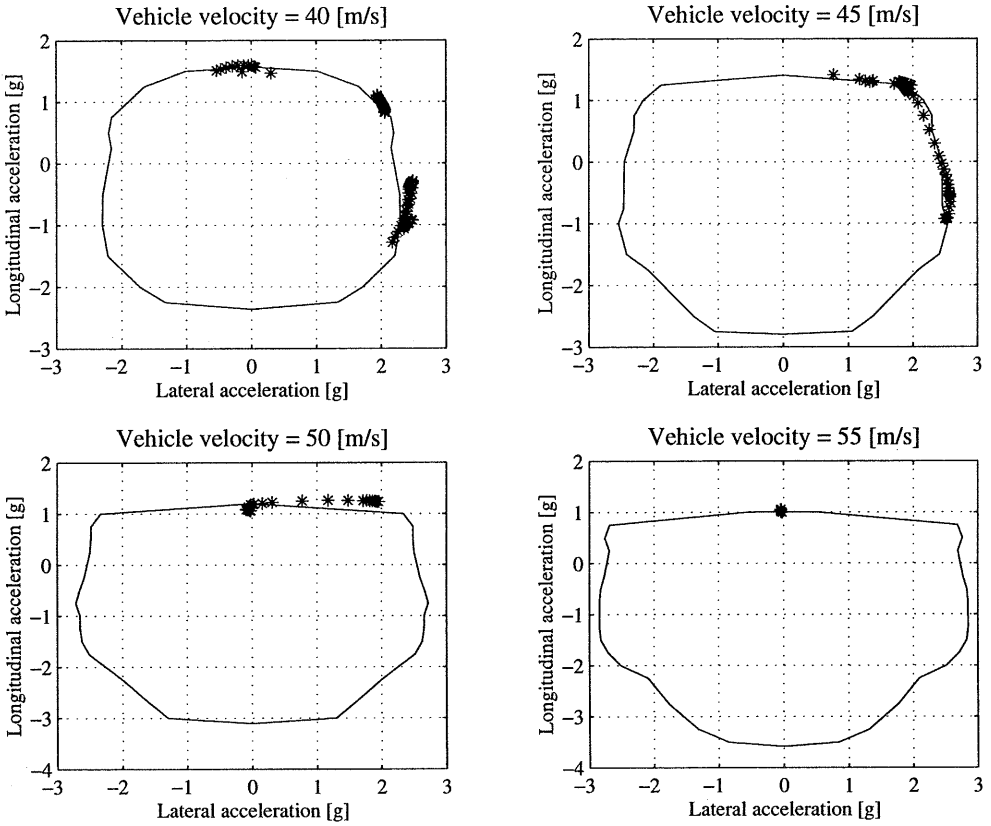


Figure 9.43 g-g analysis of the results, default yaw inertia.

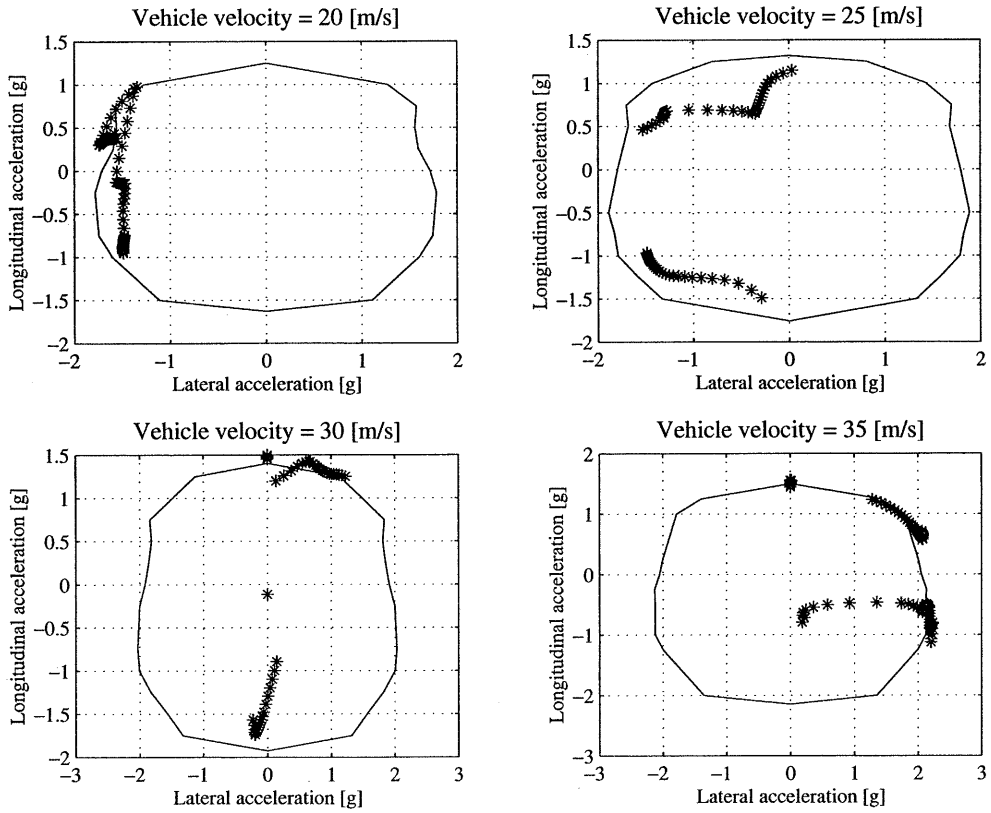


Figure 9.44 g-g analysis of the results, Izz = 10000 [kgm^2].

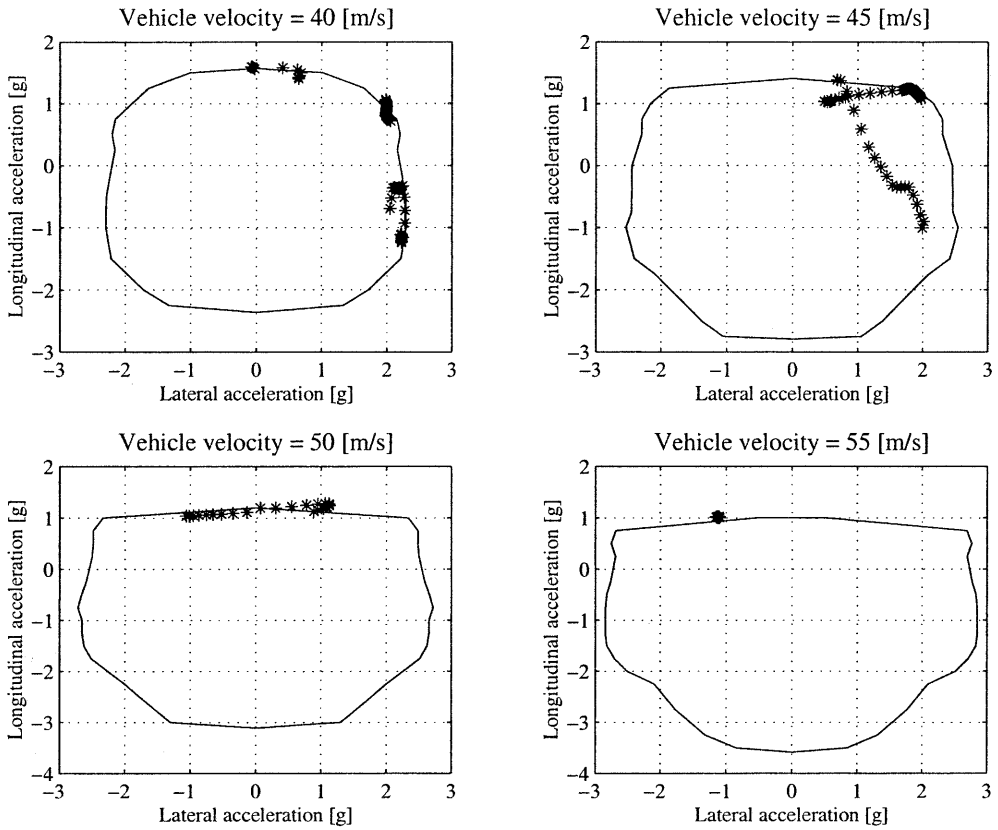


Figure 9.45 g-g analysis of the results, Izz = 10000 [kgm^2].

9.6. MASS SENSITIVITY ANALYSIS

In this section the minimum lap time for Suzuka and Barcelona is computed for a range of values of the vehicle total mass from 600 [kg] to 700 [kg], increasing 25 [kg] each time. Each case is solved four times from different trial solutions. The solutions obtained for the three precision levels for FastLap are considered. Table 9.6 summarises the results, particularly the best and worst lap time and the average number of iterations required to solve each case at different precision levels. The lower accuracy level allows to reduce the number of iterations to about one third compared to the higher accuracy level, but usually yields a greater gap between best and worst lap times.

Figures 9.46, 9.47 and 9.48 show the best lap times as functions of the vehicle mass for the three tolerance settings for FastLap. Linear interpolation is used to estimate the sensitivity of the performance. No significant difference is noticeable when considering the best solution out of four. The values of the sensitivities are nearly the same as well, and the values read 0.0369 and 0.0359 [s/kg] in high precision, 0.0362 and 0.0365 [s/kg] in normal precision and 0.0364 and 0.0366 [s/kg] in low precision for Barcelona and Suzuka respectively. Figures 9.49 and 9.50 show the gap between the best and the worst lap times for the three precision levels for each case. Even though the best lap times obtained in low precision still allow to estimate the performance sensitivity rather accurately, the uncertainty for a single lap time evaluation is much greater compared to the case of high precision. In other words, in low precision mode one would need more lap time evaluations for the same case in order to validate the

results.

Figures 9.51 to 9.56 show the vehicle controls and velocity profiles for Suzuka and Barcelona respectively. The solutions for the lightest and the heaviest vehicles are compared. The steer angle does not vary significantly, since the car balance remains the same for the lightest and the heaviest car. The longitudinal control varies essentially in braking, where the heaviest car needs to slow down earlier. The heaviest vehicle loses time in the straight sections due to the lower acceleration capability. It also loses speed in the fast corners, as both vehicles have the same downforce but the heaviest has the greatest inertial resistance to changing direction.

MASS SENSITIVITY ANALYSIS RESULTS SUMMARY							
		Barcelona			**Suzuka**		
Vehicle mass	FastLap tol.	N. iter.	Lap times		N. iter.	Lap times	
			Best	Worst		Best	Worst
600 kg	High	141	1' 18" 768	1' 19" 159	154	1' 34" 216	1' 34" 256
	Normal	73	1' 18" 894	1' 19" 481	128	1' 34" 259	1' 34" 340
	Low	72	1' 18" 898	1' 19" 481	117	1' 34" 261	1' 34" 462
625 kg	High	190	1' 19" 760	1' 19" 945	147	1' 35" 200	1' 35" 484
	Normal	128	1' 19" 861	1' 20" 197	133	1' 35" 200	1' 35" 514
	Low	111	1' 19" 904	1' 20" 325	124	1' 35" 200	1' 35" 824
650 kg	High	173	1' 20" 714	1' 20" 871	86	1' 36" 145	1' 36" 305
	Normal	118	1' 20" 834	1' 21" 003	82	1' 36" 159	1' 36" 305
	Low	81	1' 20" 946	1' 21" 080	69	1' 36" 258	1' 36" 409
675 kg	High	159	1' 21" 614	1' 21" 705	137	1' 36" 959	1' 37" 319
	Normal	113	1' 21" 664	1' 21" 818	107	1' 36" 959	1' 37" 419
	Low	110	1' 21" 682	1' 21" 853	101	1' 36" 959	1' 37" 459
700 kg	High	148	1' 22" 455	1' 22" 633	146	1' 37" 822	1' 37" 940
	Normal	114	1' 22" 513	1' 22" 751	105	1' 37" 941	1' 38" 030
	Low	103	1' 22" 553	1' 22" 758	94	1' 37" 955	1' 38" 128

Table 9.6 Mass sensitivity results, lap times and average number of iterations for high, normal and low precision mode for FastLap.

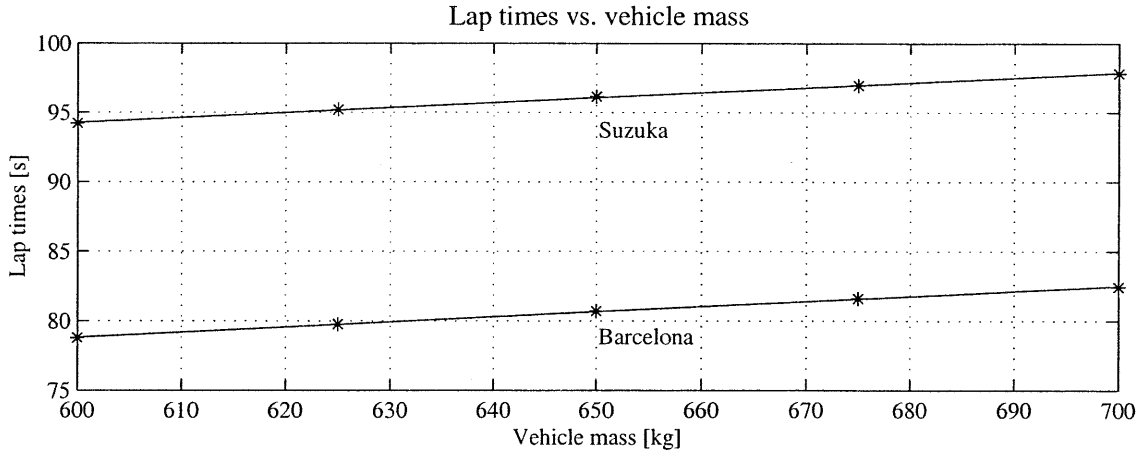


Figure 9.46 Best lap times vs. vehicle mass, results obtained with FastLap in high precision mode.

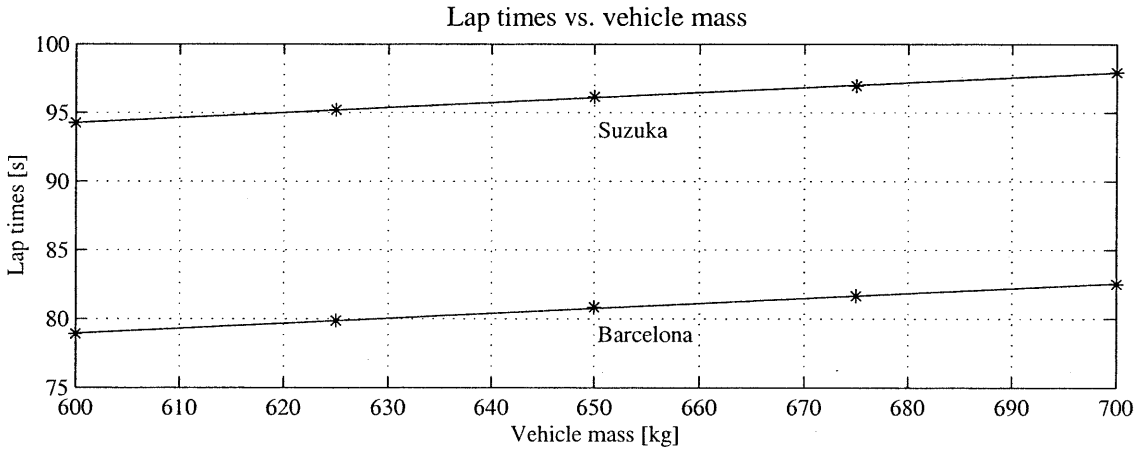


Figure 9.47 Best lap times vs. vehicle mass, results obtained with FastLap in normal precision mode.

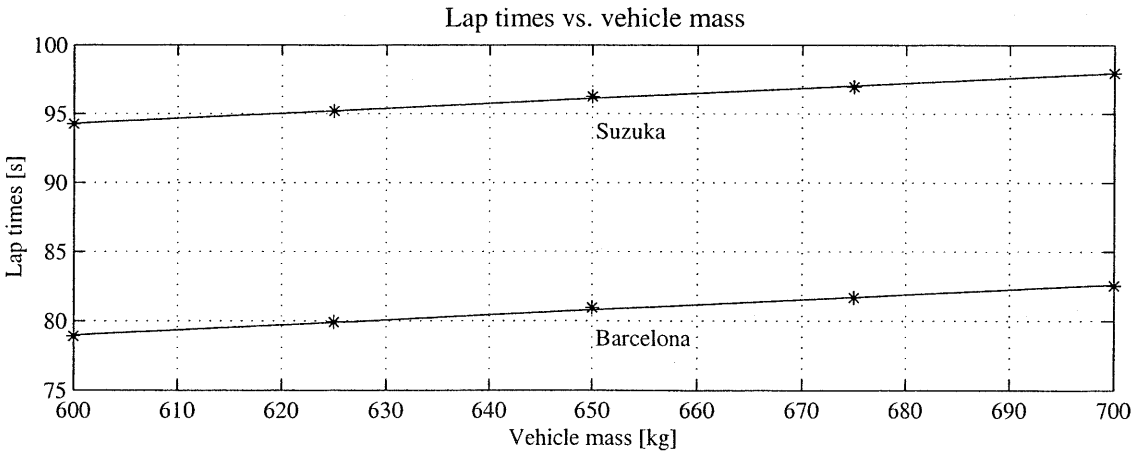


Figure 9.48 Best lap times vs. vehicle mass, results obtained with FastLap in low precision mode.

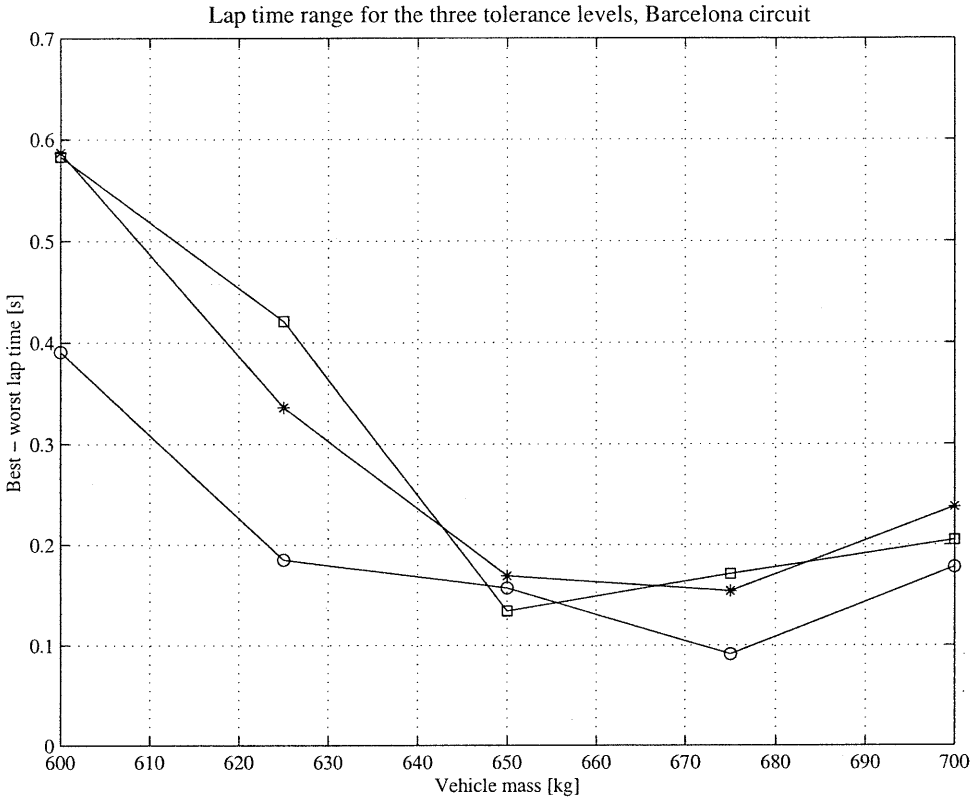


Figure 9.49 Barcelona lap times range, high (o), normal (*) and low precision (□).

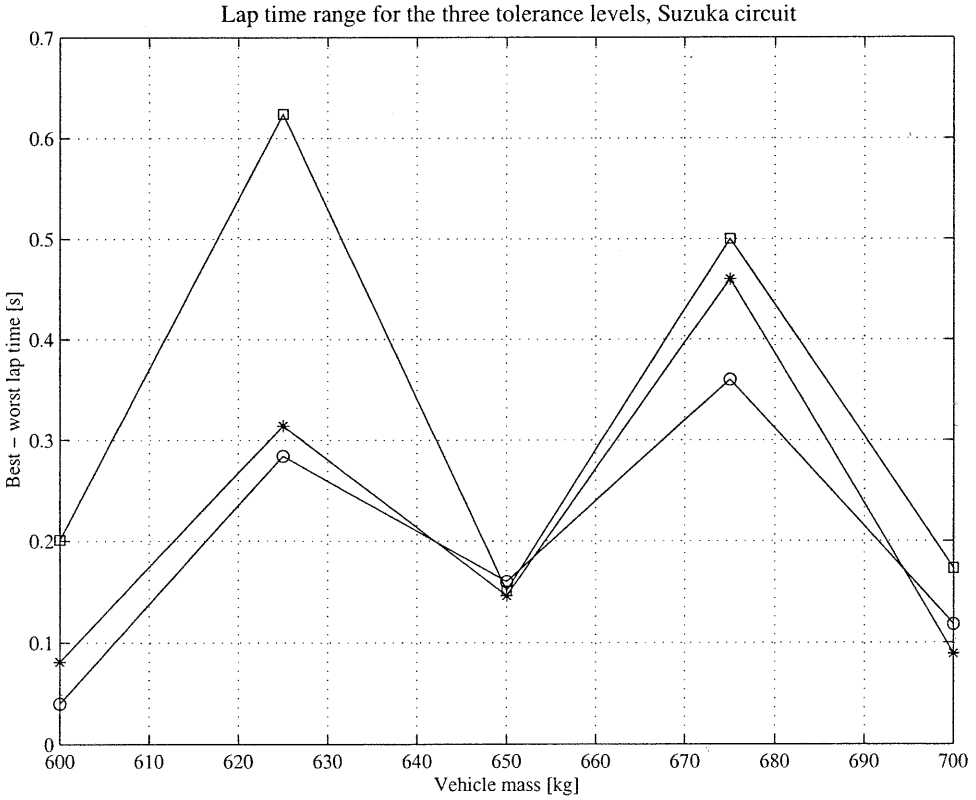


Figure 9.50 Suzuka lap times range, high (o), normal (*) and low precision (□).

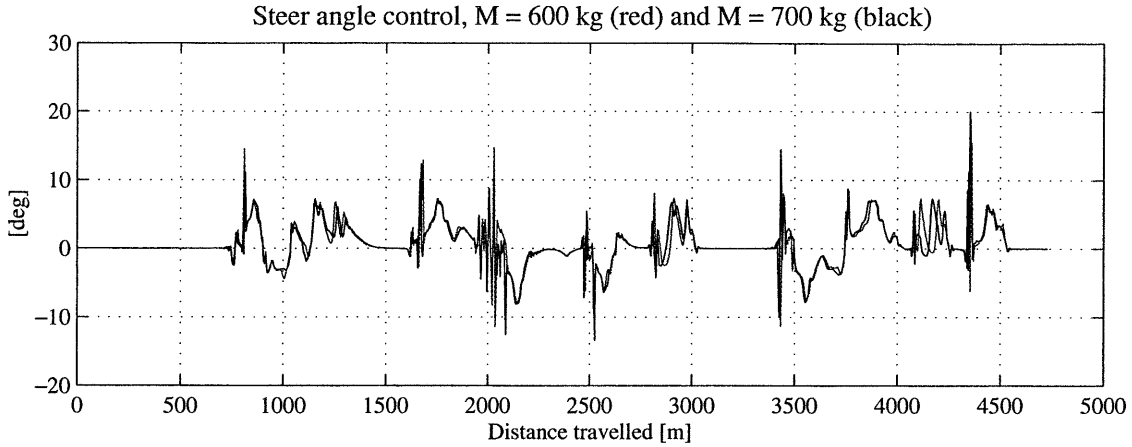


Figure 9.51 Steer angle control, Barcelona, M = 600 kg (red) and M = 700 kg (black).

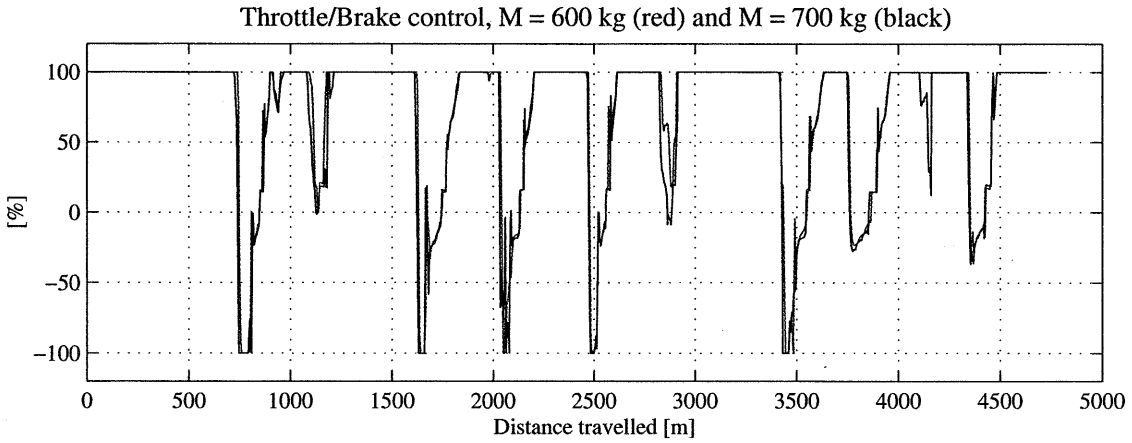


Figure 9.52 Throttle and brake control, Barcelona, M = 600 kg (red) and M = 700 kg (black).

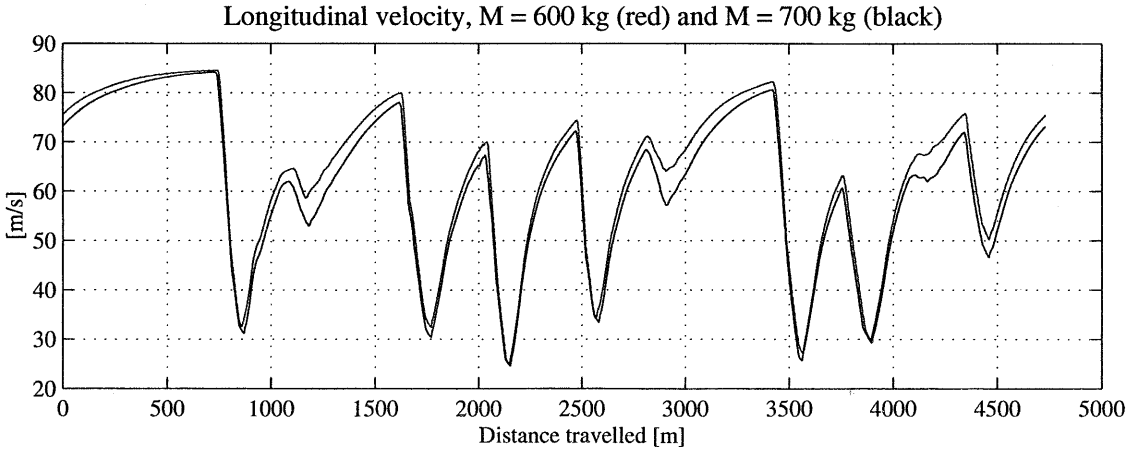


Figure 9.53 Vehicle longitudinal velocity, Barcelona, M = 600 kg (red) and M = 700 kg (black).

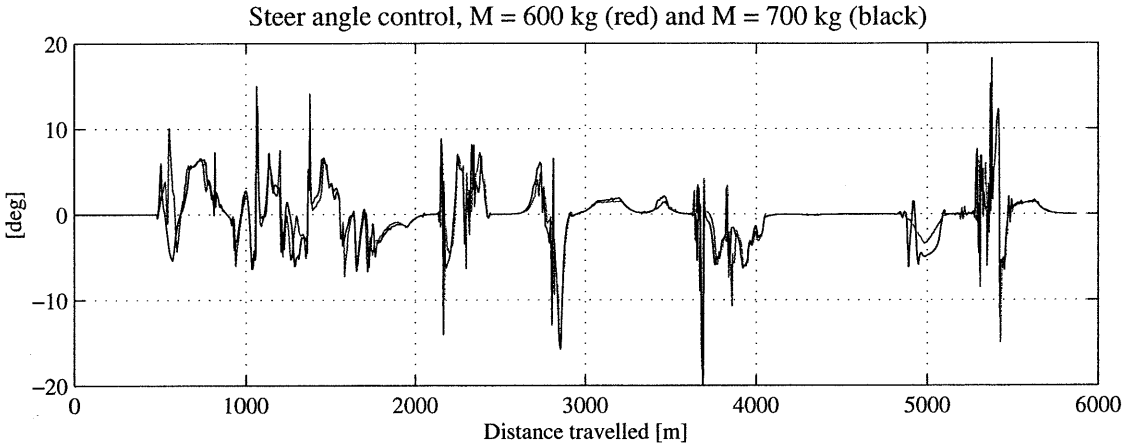


Figure 9.54 Steer angle control, Suzuka, M = 600 kg (red) and M = 700 kg (black).

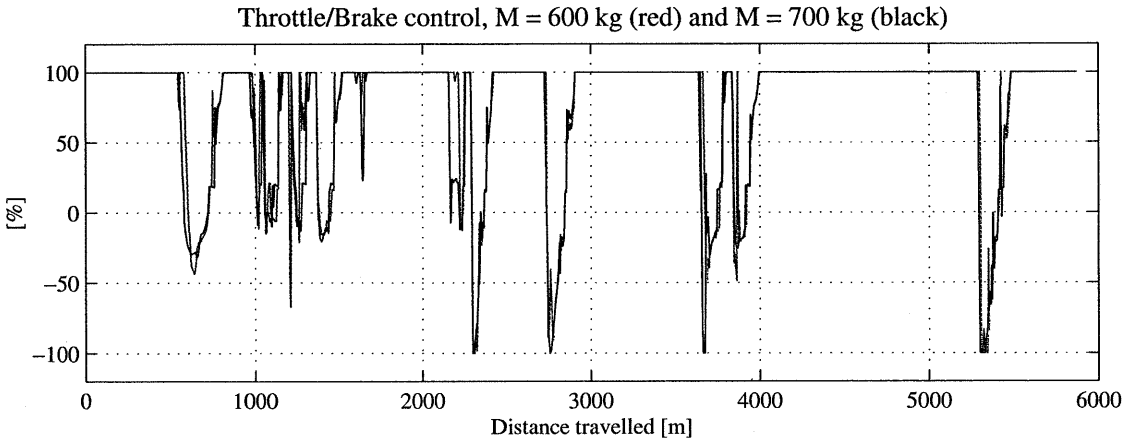


Figure 9.55 Throttle and brake control, Suzuka, M = 600 kg (red) and M = 700 kg (black).

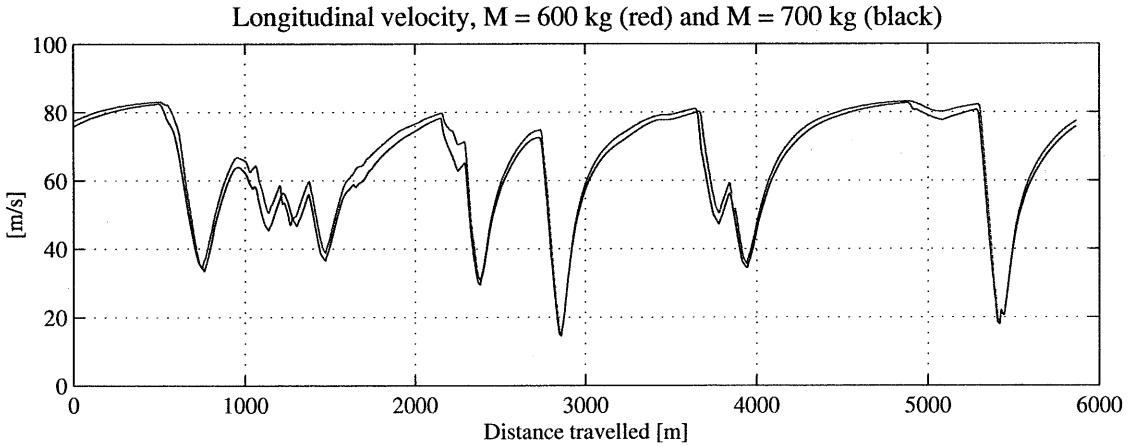


Figure 9.56 Vehicle longitudinal velocity, Suzuka, M = 600 kg (red) and M = 700 kg (black).

9.7. MASS DISTRIBUTION SENSITIVITY ANALYSIS

In this final section we investigate the optimal vehicle weight distribution which yields the best possible lap time. Nine different positions for the centre of mass, 5 [cm] far apart from each other, are considered. The resulting front and rear weight distributions are reported in table 9.7. The default values for Barcelona and Suzuka are highlighted in bold. As before, each case has been repeated four times from different initial trial solutions. Only the high precision results are considered here.

Table 9.7 summarises the results. For both circuits the results indicate that moving the centre of mass considerably rearwards yields significantly faster lap times. The best lap times are achieved by shifting the centre of mass as much as 15 [cm] rearwards for Barcelona and 20 [cm] rearwards for Suzuka. Figures 9.57 and 9.58 show the lap times against the vehicle weight distribution. The figures also show the average number of iterations which are necessary for each case. The trend is clear, the more the vehicle becomes oversteer biased, the more difficult it is for the optimisation program to converge.

MASS DISTRIBUTION SENSITIVITY ANALYSIS RESULTS SUMMARY							
Barcelona				**Suzuka**			
Weight front/rear %	No. of iter.	Lap time		Weight front/rear %	No. of iter.	Lap time	
		Best	Worst			Best	Worst
46.6/53.4	137	1' 21" 347	1' 21" 476	47.1/52.9	114	1' 36" 826	1' 37" 034
45/55	118	1' 21" 113	1' 21" 199	45.5/54.5	87	1' 36" 560	1' 36" 808
43.5/56.5	173	1' 20" 714	1' 20" 871	**44/56**	86	1' 36" 145	1' 36" 305
42/58	146	1' 20" 510	1' 20" 660	42.5/57.5	188	1' 35" 769	1' 35" 871
40.5/59.5	176	1' 20" 303	1' 20" 401	40.9/59.1	159	1' 35" 530	1' 35" 686
38.9/61.1	200	1' 20" 149	1' 20" 186	39.4/60.6	190	1' 35" 336	1' 35" 468
37.4/62.6	200	1' 20" 177	1' 20" 185	37.9/62.1	190	1' 35" 274	1' 35" 394
35.9/64.1	198	1' 20" 259	1' 20" 356	36.4/63.6	200	1' 35" 364	1' 35" 557
34.3/65.7	200	1' 20" 330	1' 20" 423	34.8/65.2	200	1' 35" 396	1' 35" 641

Table 9.7 Optimal weight distribution analysis results summary.

In order for a car to be stable and easier to control, it is necessary to have spare capacity available from the rear tyres at the limit. This implies that the vehicle cornering capability is limited by the front wheels and the full lateral force which the tyres could produce is not utilised. Moving the centre of mass rearwards changes the car balance towards oversteer, but allows to exploit the full lateral tyre forces available. However, the default set-up for the vehicle model employed here tends to saturate the rear tyres nearly at the same time as the front tyres, as shown earlier in this chapter. Yet, the optimisation program finds a considerable improvement in lap time and returns also a rather different driving strategy.

Figures 9.62 to 9.78 show the control inputs, the vehicle yaw rate, longitudinal and lateral velocity, slip angle difference and tyre utilisation index for both Barcelona

and Suzuka. The optimal solutions for the slowest and the fastest cars are compared. The rear heavy car gains speed at all corners, especially the fast ones. It also requires much less steer angle input compared with the front heavy car due to its oversteer nature. Perhaps the most significant plots regard the vehicle yaw rate and the vehicle lateral velocity. The rear heavy car goes through the corners consistently with large lateral velocity, while the front heavy car has normally a lower side velocity, which occasionally grows rapidly. This suggests that the rear heavy car, though oversteer biased, has a more predictable behaviour, while the front heavy car may prove to be even more difficult to control since it may either understeer or oversteer depending on the situations. The plots of the yaw rate are also indicative, as the rear heavy car appears to be less oscillatory than the front heavy car. Finally, a close comparison of the graphs regarding the tyre utilisation indexes shows that for the rear heavy car the average lateral saturation level of the rear tyres is generally greater and more frequently equal to 100 % compared to the front heavy car.

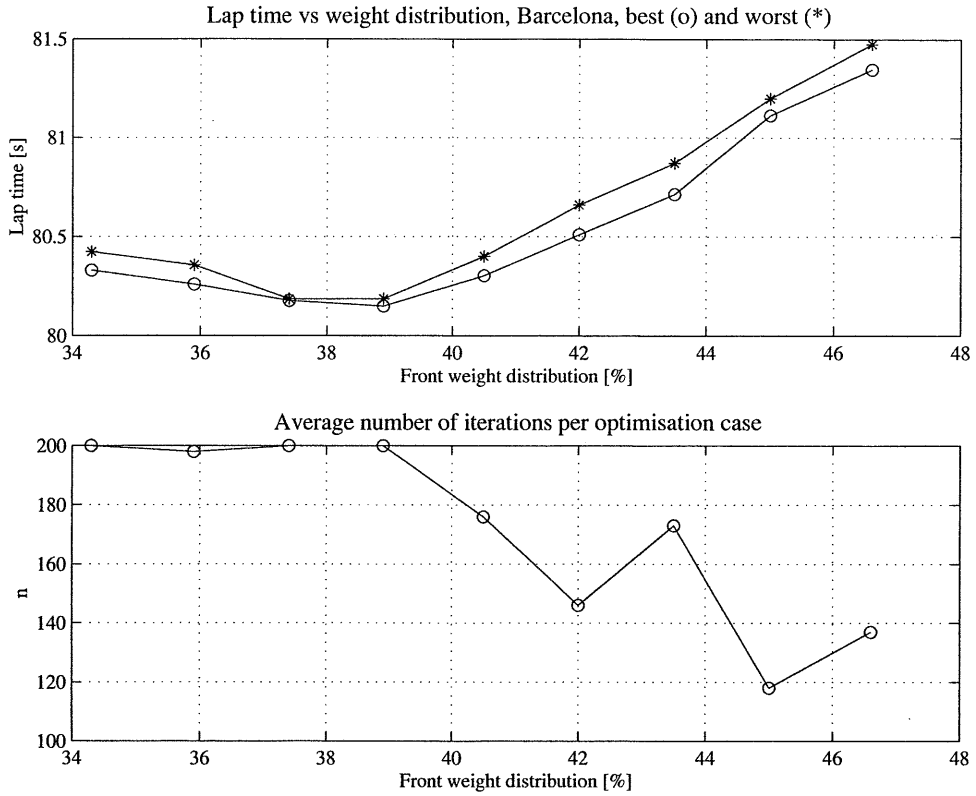


Figure 9.57 Lap time and average iterations vs. weight distribution, Barcelona.

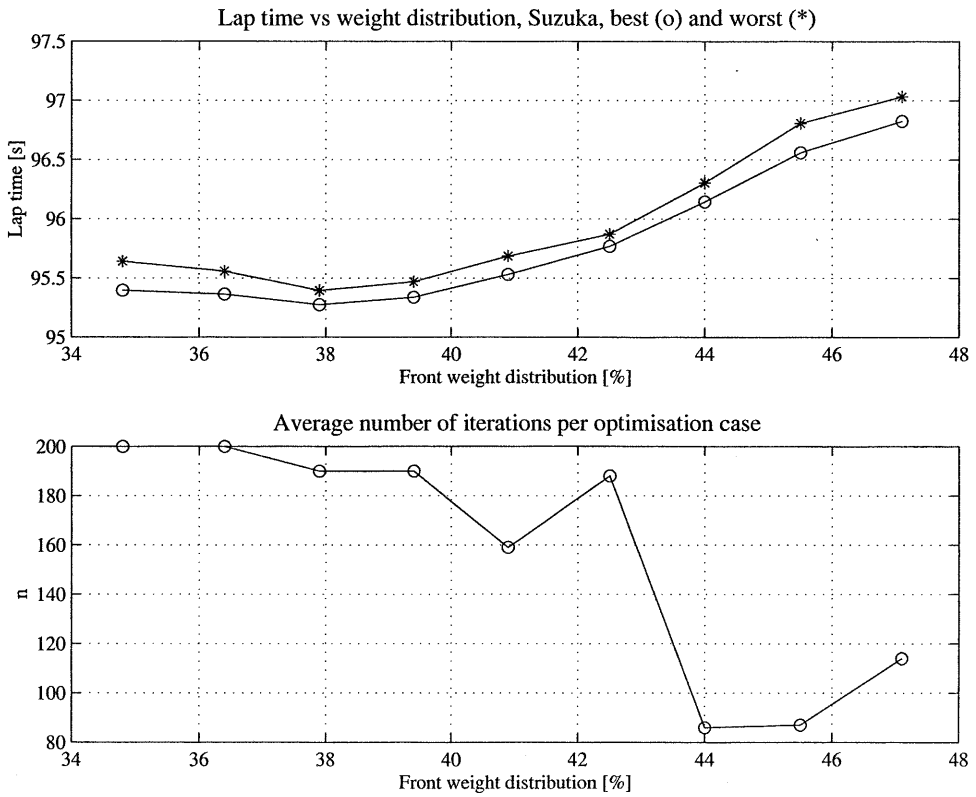


Figure 9.58 Lap time and average iterations vs. weight distribution, Suzuka.

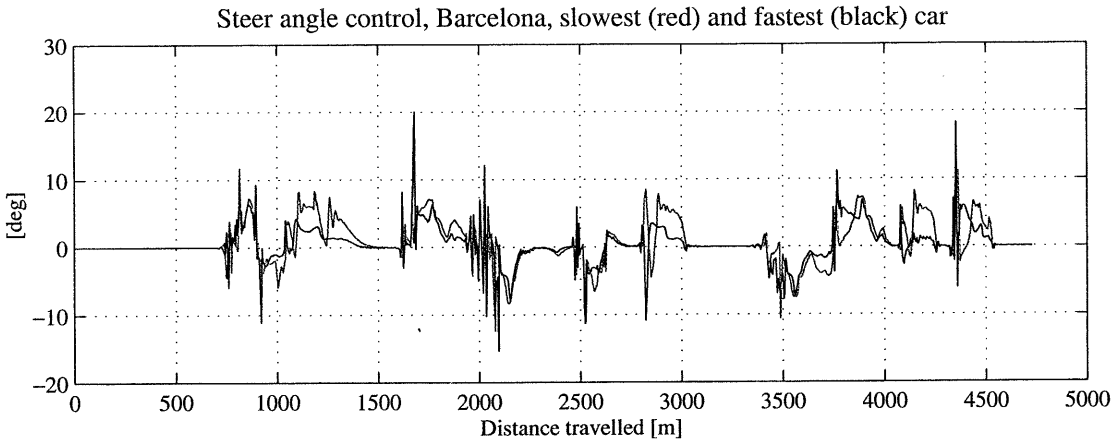


Figure 9.59 Steer angle control comparison, Barcelona.

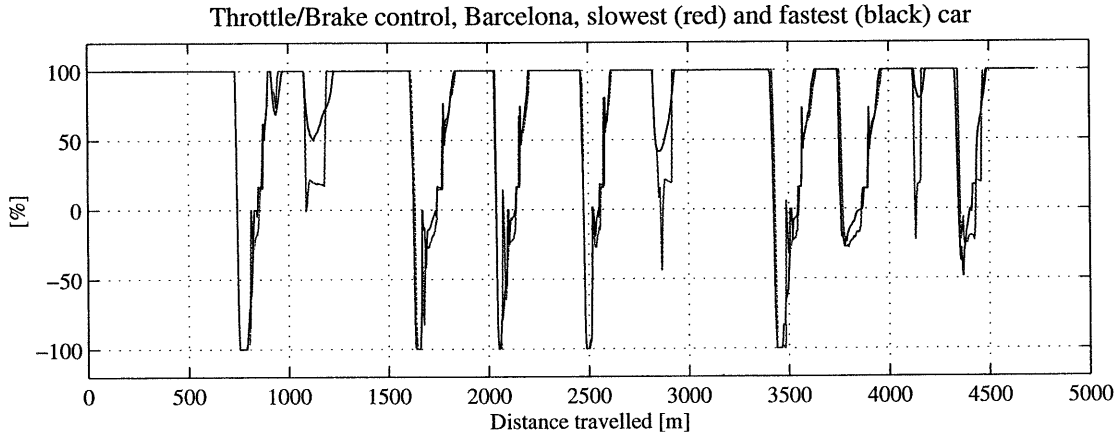


Figure 9.60 Throttle and brake control comparison, Barcelona.

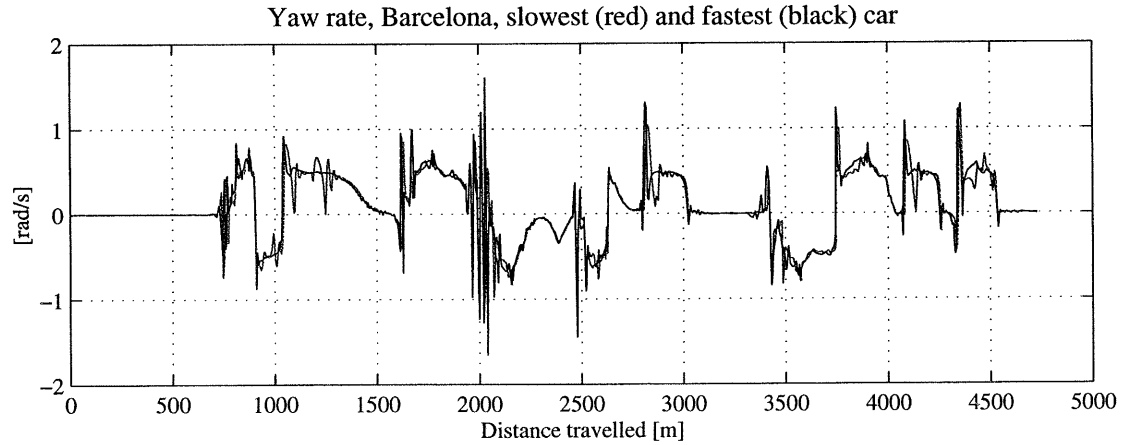


Figure 9.61 Vehicle yaw rate comparison, Barcelona.

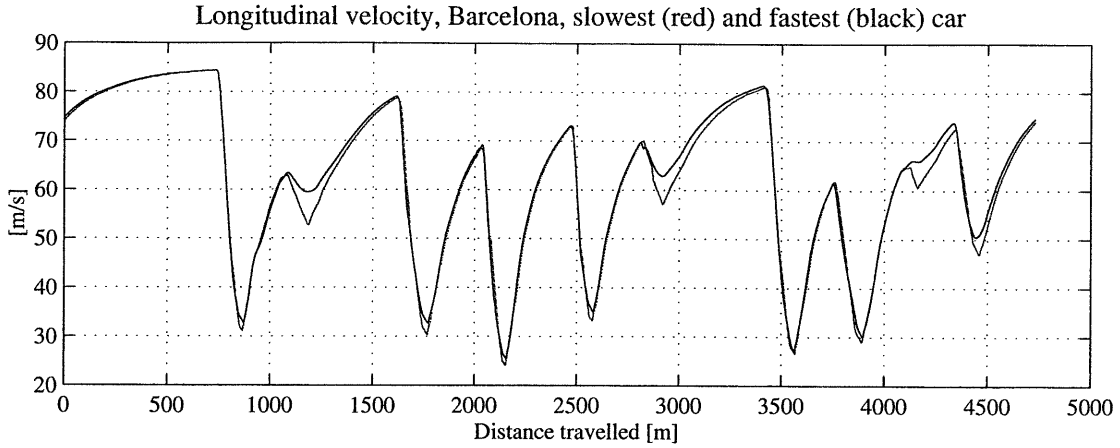


Figure 9.62 Vehicle longitudinal velocity comparison, Barcelona.

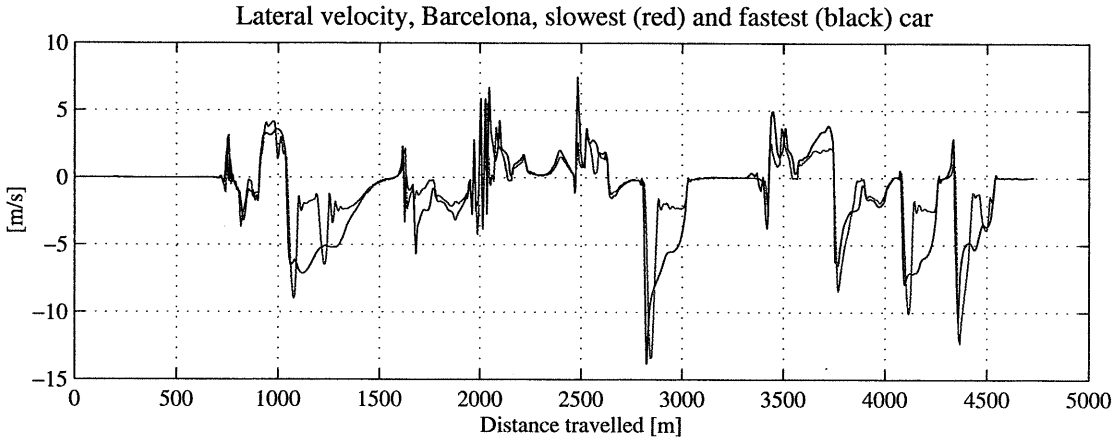


Figure 9.63 Vehicle lateral velocity comparison, Barcelona.

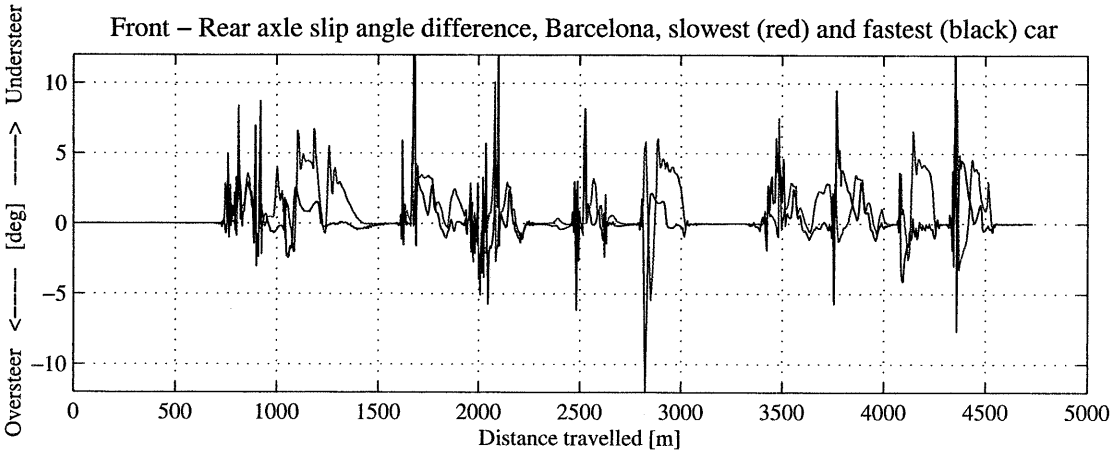


Figure 9.64 Front and rear slip angle difference comparison, Barcelona.

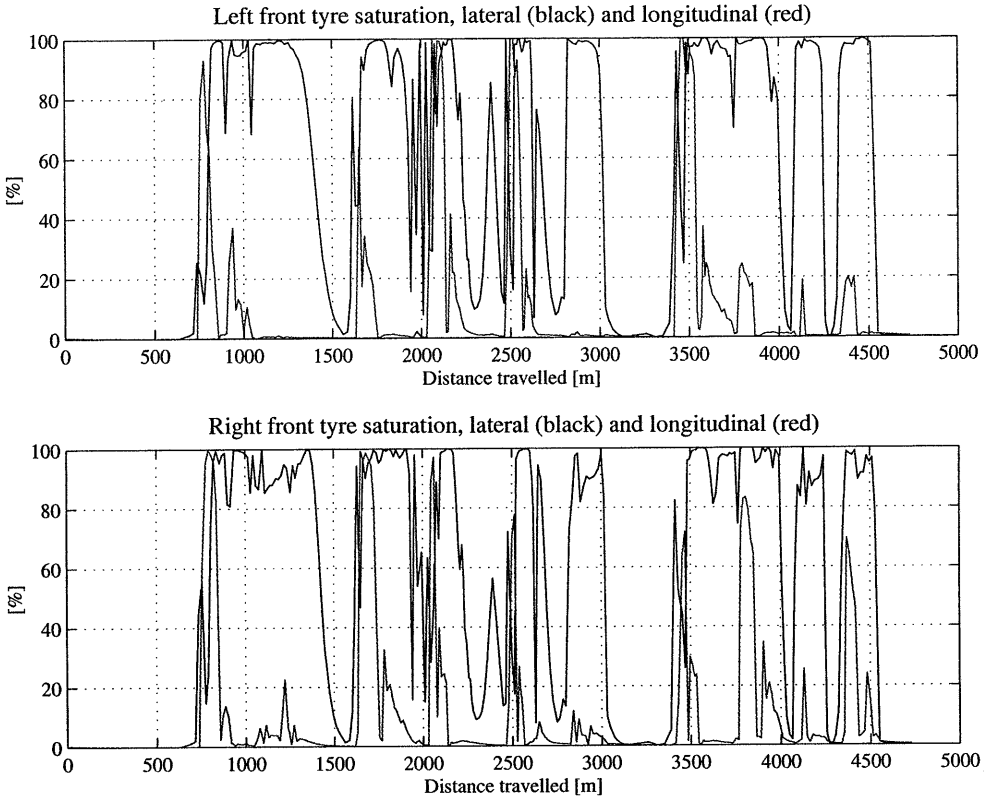


Figure 9.65 Front wheels tyre utilisation index, slowest car, Barcelona.

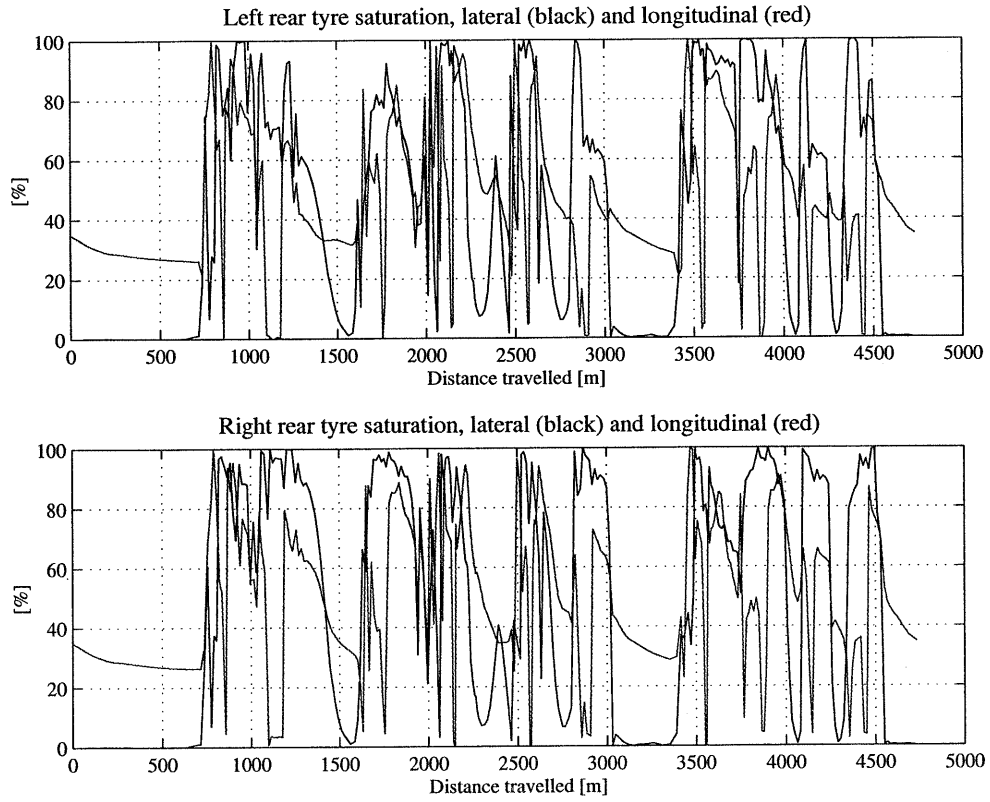


Figure 9.66 Rear wheels tyre utilisation index, slowest car, Barcelona.

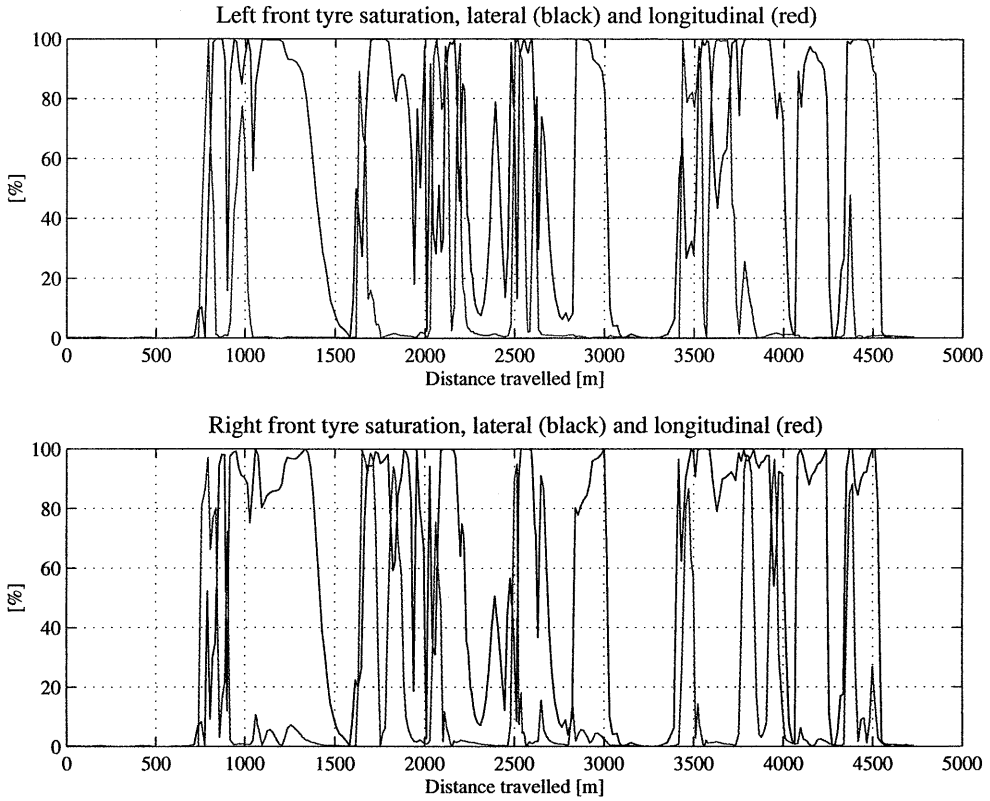


Figure 9.67 Front wheels tyre utilisation index, fastest car, Barcelona.

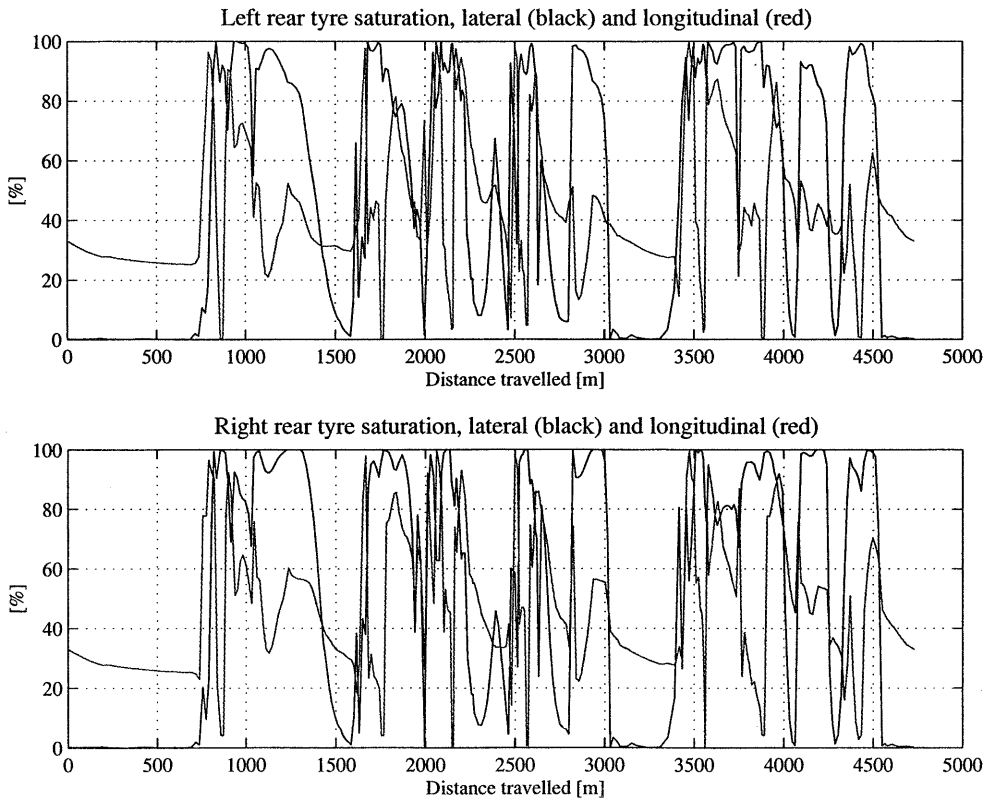


Figure 9.68 Rear wheels tyre utilisation index, fastest car, Barcelona.

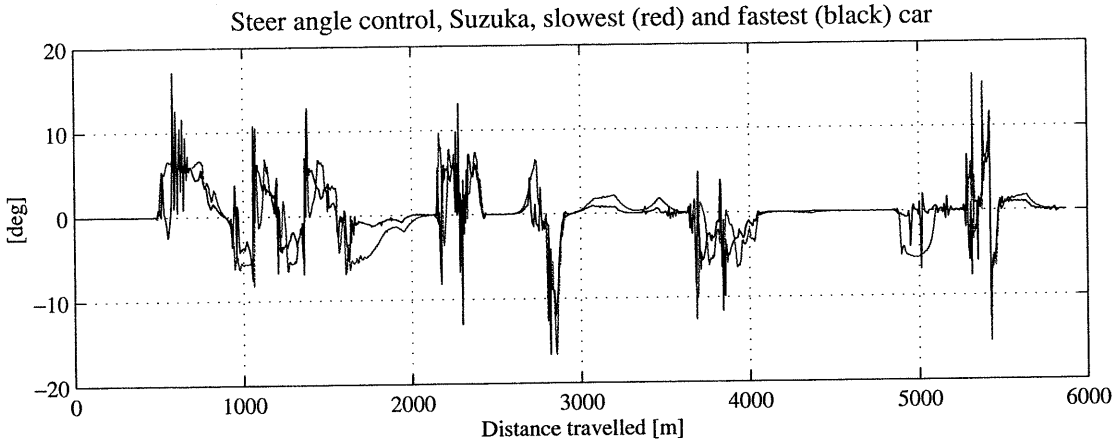


Figure 9.69 Steer angle control comparison, Suzuka.

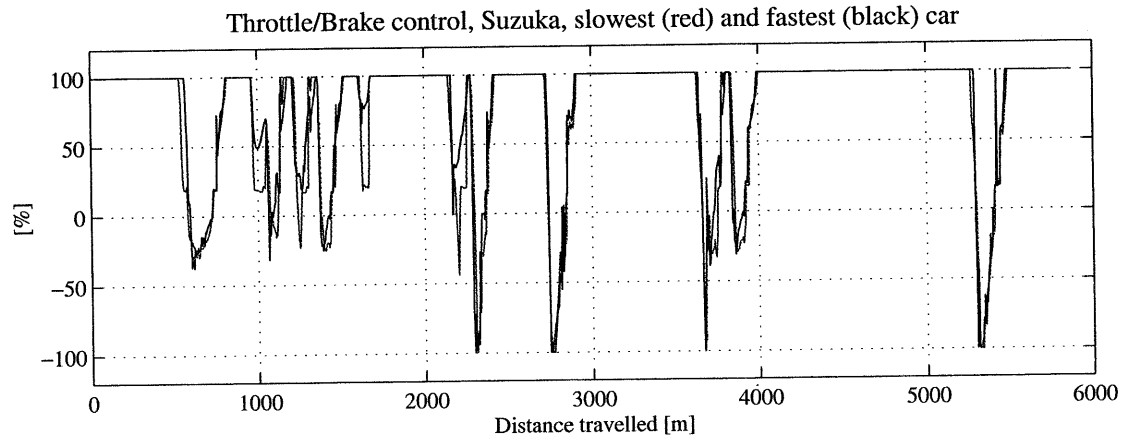


Figure 9.70 Throttle and brake control comparison, Suzuka.

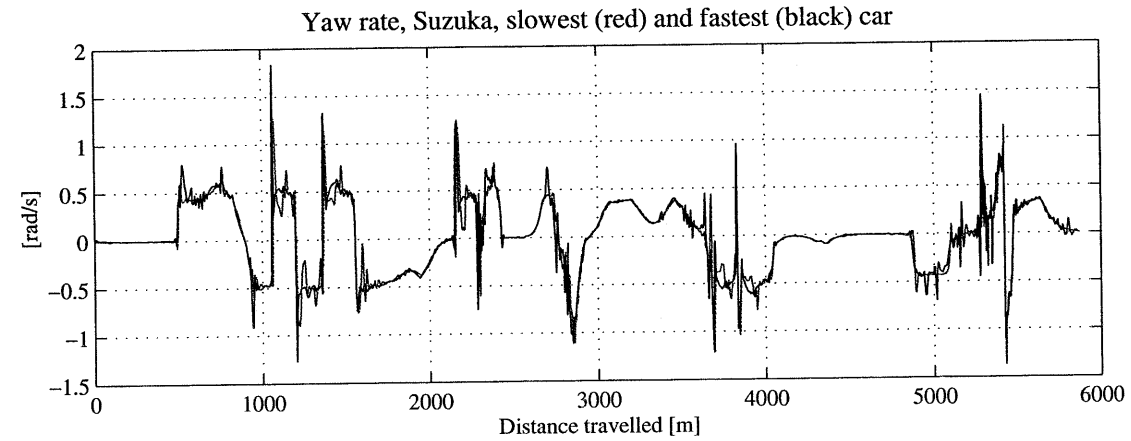


Figure 9.71 Vehicle yaw rate comparison, Suzuka.

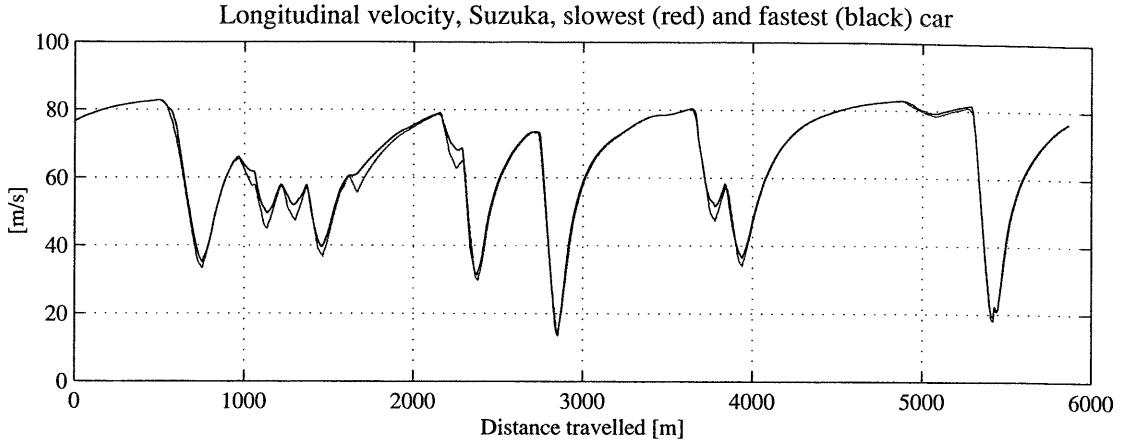


Figure 9.72 Vehicle longitudinal velocity, Suzuka.

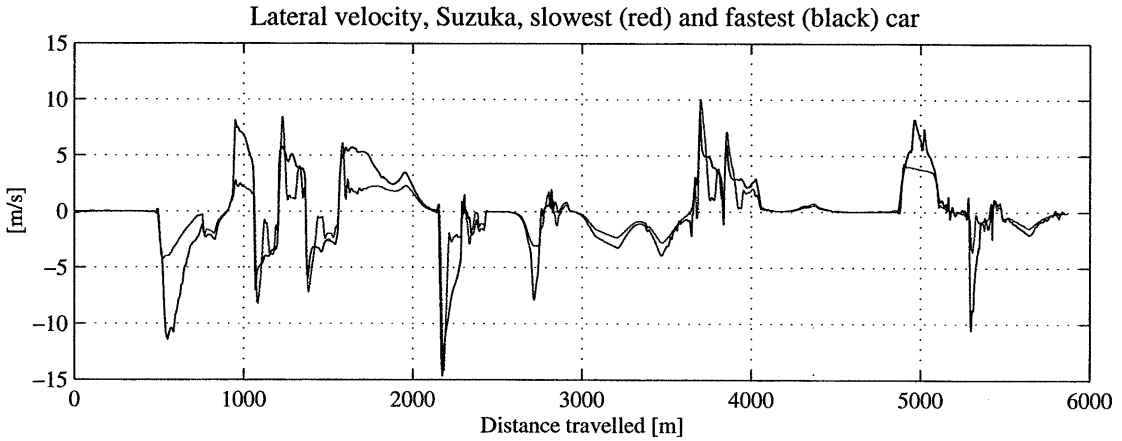


Figure 9.73 Vehicle lateral velocity, Suzuka.

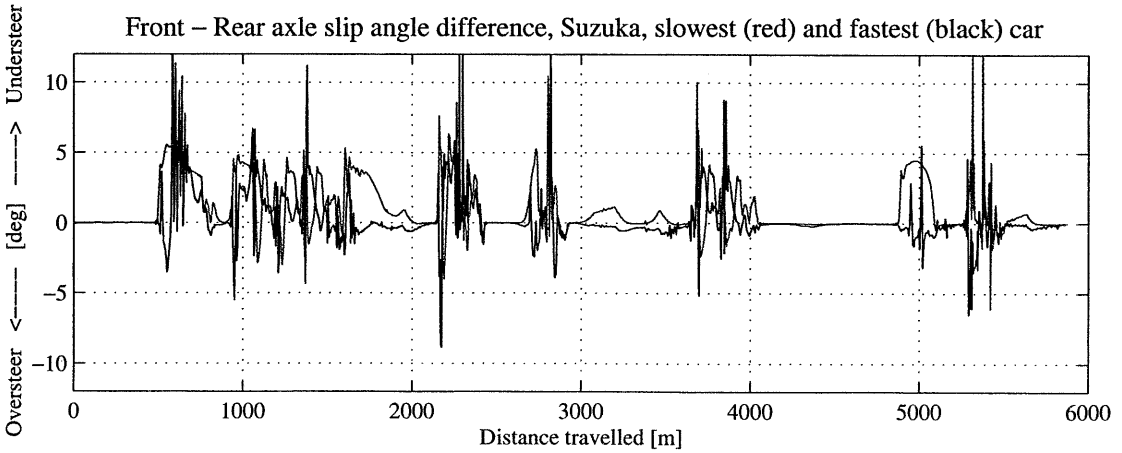


Figure 9.74 Front and rear slip angle difference comparison, Suzuka.

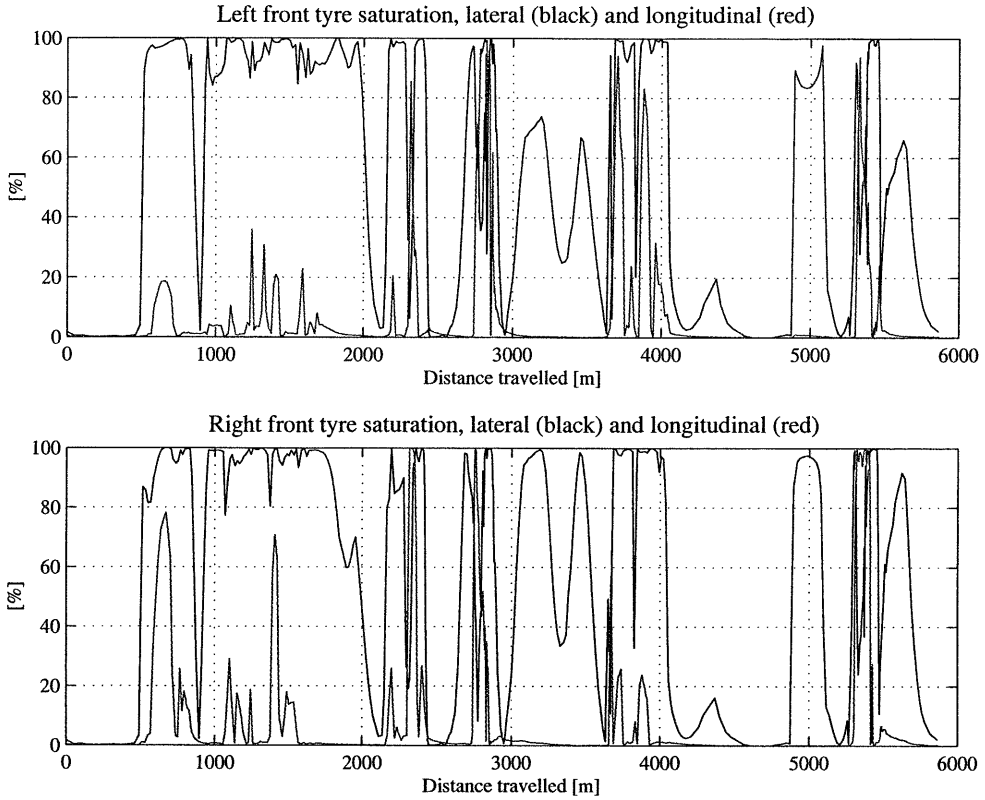


Figure 9.75 Front wheels tyre utilisation index, slowest car, Suzuka.

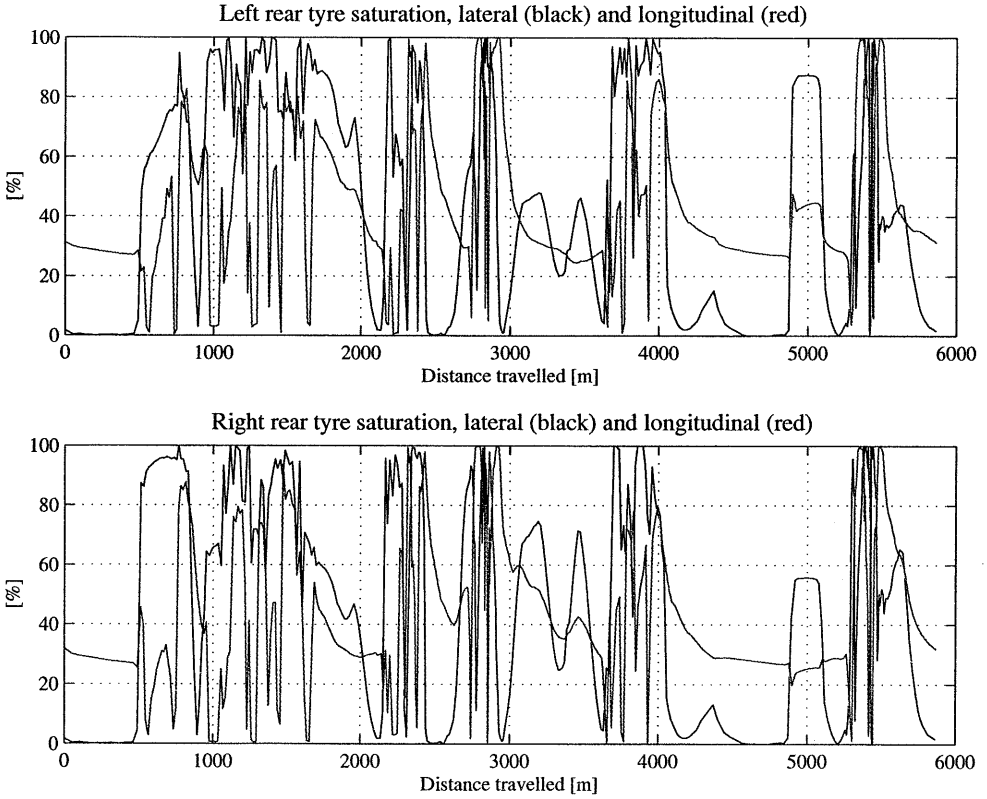


Figure 9.76 Rear wheels tyre utilisation index, slowest car, Suzuka.

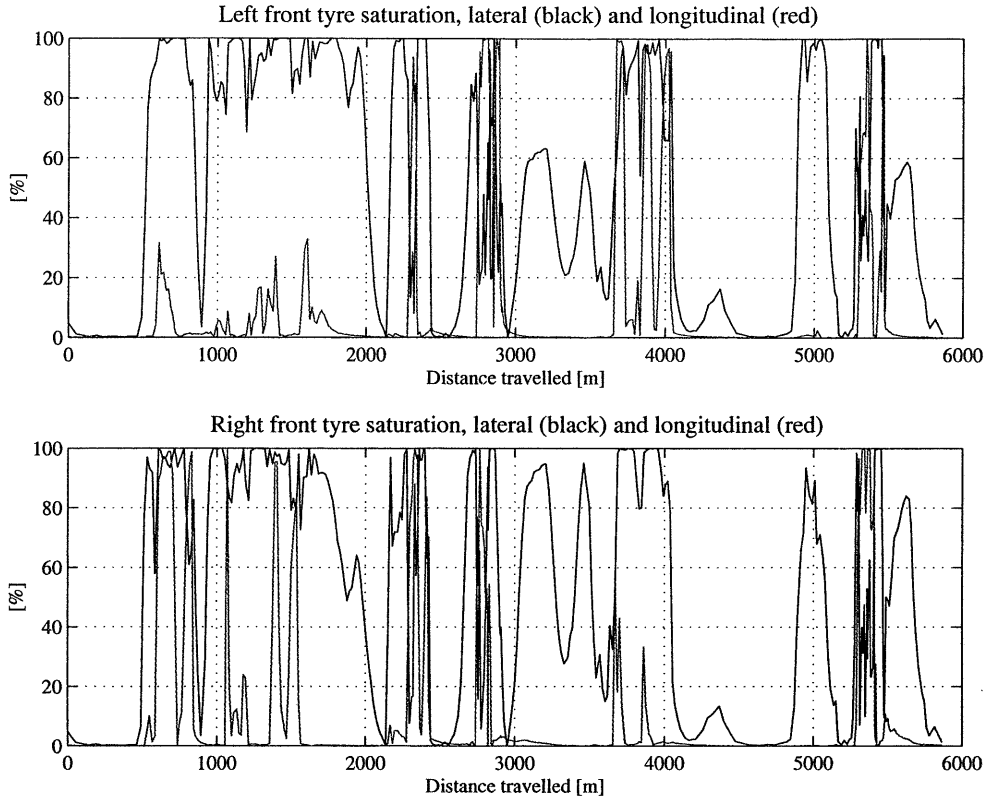


Figure 9.77 Front wheels tyre utilisation index, fastest car, Suzuka.

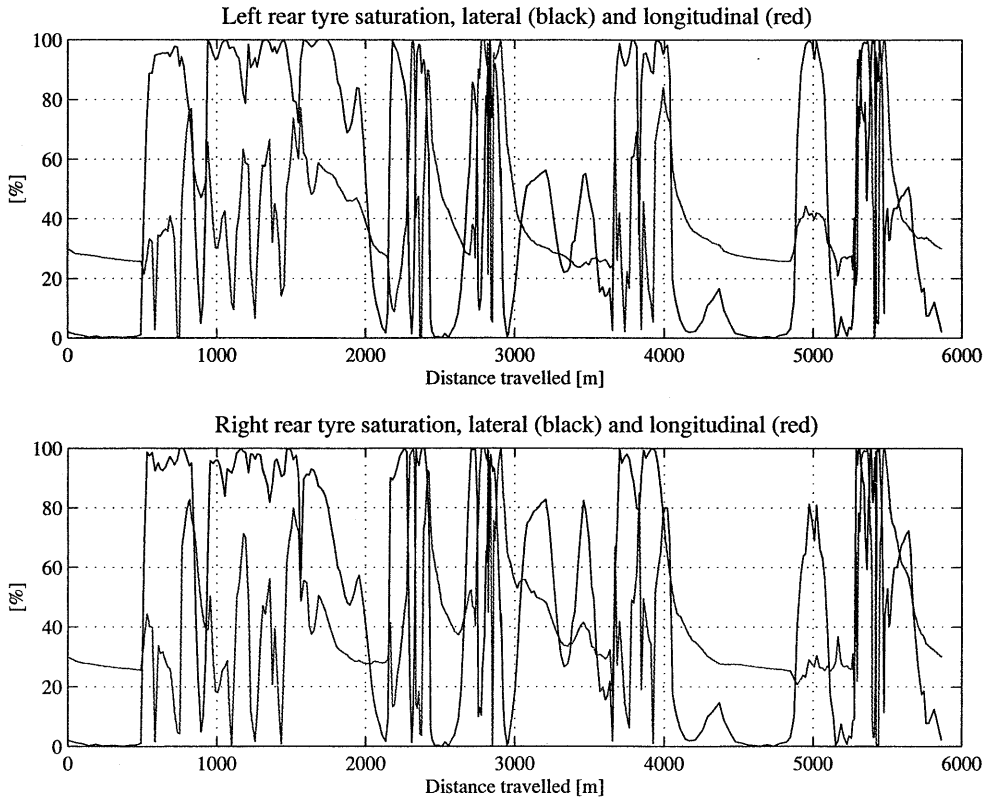


Figure 9.78 Rear wheels tyre utilisation index, fastest car, Suzuka.

9.8. CONCLUSIONS

The lap time optimisation program FastLap has been applied to a variety of study cases involving different track geometries and vehicle configurations. The robustness of the program and the accuracy of the results has been demonstrated. Particularly, the results are always consistent with the dynamic properties of the vehicle model used.

With regard to the yaw moment of inertia, a common concept in race car design is to concentrate the masses as much as possible near the centre of gravity. However, modern Formula One cars can generate very large tyre lateral forces thanks to the aerodynamic downforce. This fact and the usually very long wheelbase allow to generate a very large moment about the vehicle yaw axis. Hence, it is not unexpected to find that the sensitivity of the performance with respect to yaw inertia is negligible for a very large range of inertias. At this stage the controllability issue appears more important, as it appears that increasing the yaw moment of inertia makes the car easier to control without penalising the performance.

The mass sensitivity problem is of great importance in current Formula One races because of the strategy involved by the compulsory pit-stop for re-fuelling. Racing strategy has become an integral part of the Formula One show. Teams usually perform lots of tests to identify how the variation of the fuel load affects performance, and these results are known to the general public to some extent. In general, the practical results indicate that the sensitivity of the lap times with respect to variations of the vehicle mass is of the order of 0.04 second per kilogram of fuel, and this is in very good agreement with the simulations presented here.

Finally, the weight distribution problem is also very important nowadays. The latest engine and chassis technology allows to build cars which are significantly under the minimum weight allowed. As a result, the teams have a considerable amount of ballast on the car which allows significant variations of the weight distribution. The results presented here are as well in line with the practical experience. However, the problem on the track is more difficult since the real drivers may find a car which is potentially faster almost impossible to drive. On the other hand, it is well known that very successful drivers are able to control cars which are more oversteer biased.

With regard to the accuracy, the initial goal of evaluating the minimum lap time within one millisecond of accuracy has not been met. The current level of accuracy of FastLap, though, appears to be sufficient to quantify the sensitivity of the vehicle performance to parameter variations. The spread of the results obtained from different runs for the same study case may actually be due to local minima. In fact, the nature of the minimum time vehicle manoeuvring problem is in such a way that it is often possible to trade off velocity in one corner for more speed through the next corner. On the other hand, the progressive development of the FastLap in terms of accuracy has allowed to shrink the range of values in which a solution may fall, and the ideal strategy seems to be to try to further lower the tolerances in order to rely on one single lap time evaluation per study case, rather than accepting poor accuracy and performing repeated runs.

In summary, the results presented here show the potential of the lap time optimisation program to analyse the vehicle transient and limit behaviour. Nevertheless, more work is needed, mainly to further enhance the accuracy and to reduce the large computational times. In the next, conclusive section we shall address all the necessary issues to make FastLap the next generation of lap time optimisation tool.

Chapter 10.

Conclusions and Further Work

The natural extension of simulation is optimisation. CAO, Computer Aided Optimisation, is the latest acronym among the Computer Aided techniques which have revolutionised engineering design in the past decades. The core technology for optimisation, though, is by no means new. For example, algorithms for constrained or unconstrained minimisation of general non-linear functions have been under development for several years. The earliest implementations of Automatic Differentiation date back to the late seventies. However, the growth of computer power and the development of modern programming languages now make optimisation techniques more efficient, robust, user friendly and ultimately attractive for engineering applications.

Vehicle modelling is a well established subject. Current models allow to analyse and predict the vehicle transient behaviour with good accuracy. The ideal development is to have a formal procedure to evaluate the optimal control strategy and the optimal combination of the design parameters to achieve a certain goal. The difficulty is often to describe such goal mathematically, in such a way that the problem can be solved by using a formal optimisation method. If we think, for example, of a road car, its handling and ride qualities have to be optimised in a full range of conditions and speeds. Furthermore, a road car is driven by many people with different skills and attitudes. Hence the optimisation process can hardly be de-coupled from human factors. The problem in racing is much simpler to a great extent. The task is to traverse laps of a circuit in as short a time as possible. The car has to be driven all the time as close as possible to its performance limits. Yet, human factors are involved, since a human driver, though super-skilled, is in control of the car.

A general method for the evaluation of the theoretical optimal lap for a transient vehicle model has been presented in this work. The problem is formulated as one of Optimal Control theory and is solved using mathematical programming techniques. The nature of the minimum time vehicle manoeuvring problem has been investigated and exploited in the development of the solution strategy. State-of-the-art optimisation technology has been employed to solve the resulting large scale, non-linear constrained minimisation problem. Excellent robustness and computational efficiency have been achieved. Even though the full potential of Automatic Differentiation is not attained in the present implementation of the lap time optimisation program, its advantages have been demonstrated and future developments of FastLap will benefit more from this technology. Various application study cases have been presented and the effectiveness of the lap time optimisation method has been demonstrated. Particularly, FastLap finds the racing line, allows to simulate the race car on the edge of its performance, to quantify such performance and its sensitivity to design parameter variation, and to

investigate its limit behaviour, leaving scope for subjective judgement of its handling qualities.

In the course of this research other subjects related to the theoretical optimal lap have been considered, such as the reconstruction of the racing line from on-board measured data and the driver model. In the next paragraphs the main findings in each area will be reviewed, and the scope for further developments and new applications will be highlighted.

10.1. ON THE RACING LINE RECONSTRUCTION FROM MEASURED DATA

The problem of reconstructing the racing line from on-board measured data, i.e. vehicle longitudinal velocity and lateral acceleration, was investigated in chapter 3. Two algorithms to perform this task were developed, and both proved to be capable to generate realistic racing lines. However, the results highlighted the large sensitivity of the results with respect to errors which are inevitably present in the measured data. The kinematic model employed is also very simple, i.e. the vehicle side slip and the wheel longitudinal slip quantities are not included, and this is partially responsible for the lack of accuracy. On the other hand, the large sensitivity to inaccurate measured data seems to be the primary problem and a more elaborate kinematic model would not probably yield a significant improvement on its own. Furthermore, data such as the vehicle side slip velocity are not normally measured on the real car.

The accuracy of the current results is sufficient for the purpose of some vehicle dynamics analysis, such as path following tasks, as was shown in chapter 5. However, the initial plan of using the reconstructed racing lines as the base for deriving the initial trial solutions for the lap time optimisation procedure proved to be not convenient. The reconstructed racing lines, though very similar in shape to the related circuits when represented on their own, are usually distorted and lie too far outside the accurate maps of the tracks. The problem was overcome by setting up a procedure which involves drawing the initial racing line interactively over the map of the circuit, which was described in chapter 8. This is a relatively slow process, and is acceptable if considered a part of the modelling of the circuit and if it is to be done only once. However, when the map of the circuit is available, an alternative strategy might be possible in order to fully automate the generation of an initial racing line which lies mostly within the road boundaries.

The standard procedure to reconstruct the racing line involves computing its curvature from the vehicle longitudinal velocity and lateral acceleration. Then, by integrating the curvature twice, the map of the racing line is obtained. The double numerical integration is responsible for the large sensitivity of the results with respect to an inaccurate starting set of data, as a small error in the curvature at the beginning of the trajectory is carried on and causes an increasing drift which results in large errors at the end of the trajectory. To some extent, this relates to the problems in the lap time optimisation algorithm which arise due to the coupling between early controls and later vehicle states. As was extensively discussed, this led to the discretisation of the vehicle trajectory in short, totally de-coupled segments.

The same reasoning suggests a new strategy for the reconstruction of a racing line, suitable for the lap time optimisation program, when the map of the circuit is available. The reconstruction of the racing line would have to be divided in segments as well. The

start of each segment would be set to lie inside the road boundaries. Furthermore, the continuity and smoothness of the segments across the junctions would have to be ensured by specifying sufficient constraints. A different set of curvature compensation coefficients for each segment of the trajectory would have to be specified, using the same strategy as that of the error compensation algorithm of §3.4. A continuous trajectory could then be obtained by adjusting the coefficients for each segment independently, and the problem may be solved by employing optimisation methods, where the objective would be to close the gaps at the many segment junctions, ensuring also smoothness.

10.2. ON DRIVER MODELLING

A new structure for a steering controller for road vehicles has been devised. The controller monitors the path ahead and takes samples of path preview errors, lateral position error and attitude error. The observations are then converted into steer angle control input. The basic structure originates from Linear Optimal Preview Control Theory, but has been developed to deal with non-linear vehicle operations arising from the saturating tyre forces in vigorous manoeuvring. The implementation and tuning of the model has been described. Several path following tasks have been solved and the path tracking capabilities of the steering controller demonstrated. Further work is planned in order to improve the path following capabilities when the vehicle is near its lateral limit, the parameter tuning process and the coupling between longitudinal and lateral controls.

The current implementation of the driver model computes the steer angle as a linear function of the controller inputs by means of constant gains. Furthermore, saturation functions are used to prevent the growth of the computed steer angle when the path tracking error increases on approaching the friction limit of the tyres. Even though this has enhanced the driver model capability on limit manoeuvres, a neural network structure seems to be the natural development of the current structure. It would provide a better adaptation of the driver model response according to the non linear tyre characteristics and would also provide the possibility of an automated strategy for parameter tuning by means of neural network learning techniques.

In the current driver model lateral and longitudinal vehicle controls are treated as completely de-coupled. The longitudinal velocity is usually imposed, and if this leads to exceeding the vehicle performance boundary, the driver model can not control the car further and the simulation is stopped. In principle, the vehicle longitudinal control would need some preview information regarding the curvature of the path ahead, as well as the lateral control needs. Furthermore, it would also need some state feed-back, in order to monitor how close to its performance limits the vehicle is. However, it is not clear at the moment what the structure of such controller should be, what law should relate the longitudinal control with the input signals and how such controller could be tuned. Nevertheless, it would be of interest to have a driver model not only capable of good path following, but also capable of following a pre-defined line as fast as possible.

Even though no work has yet been done on the subject in the context of this research, it is thought that optimisation methods can be used for this purpose. The general method would have as objective the time that the vehicle takes to follow the entire length of the intended path. No road constraints would be needed, as the

procedure would rely on the steering controller in following the path. Furthermore, the steering controller has a stabilising effect for the vehicle. Hence it is likely that dividing the trajectory in segments as in FastLap will not be required. The longitudinal control would be the throttle and brake input as function of the path length, and would be discretised in the same way as in the original lap time optimisation problem. The resulting optimisation problem would consist in minimising the objective, i.e. the manoeuvre time, by adjusting the longitudinal control. No constraints would apply, but only suitable longitudinal control bounds. Such problem would be much smaller compared to the full lap time optimisation problem, and would have the added advantage that computing derivatives of a scalar valued objective function using Automatic Differentiation is very efficient.

10.3. ON LAP TIME OPTIMISATION

The lap time optimisation program FastLap is the main outcome of this research dedicated to the theoretical optimal lap. Vehicle dynamics simulation and optimisation technology have been brought together to define and solve the general problem of finding the best lap time for a datum circuit and vehicle model, without any of the simplifying assumptions which characterise the lap time simulation tools currently in use in Formula One racing. Many are the areas where improvements can be achieved, and one above all is with regard to the computational time.

The solution method chosen for the lap time optimisation problem relies on the ability to compute derivatives of a large vector-valued function of several independent variables. The difficulties in computing fast and reliable derivatives in a non-linear optimisation problem such as FastLap have been highlighted. Automatic Differentiation is seen as a key factor in improving computational efficiency, accuracy and robustness. Future versions of FastLap will have to be built around an Automatic Differentiation library which provides the necessary functionality. This is required in the vehicle modelling process, for example when using look-up tables. Furthermore, the strategy to differentiate the entire numerical integration of the vehicle dynamics equations is not ideal, as it causes a memory storage problem. This is bound to penalise efficiency when having to deal with the very large data structures created in the computer memory at run time. The alternative better strategy, suggested by (Eberhard and Bischof, 1999) involves using Automatic Differentiation to differentiate only the right-hand side of the ODE system. This procedure yields the sensitivities of the state derivatives with respect to the independent variables. These sensitivities are integrated alongside the state derivatives and this allows to obtain state trajectories as well as their sensitivities with respect to the independent variables. Since the objective is a function of the states, the chain rule applies. Furthermore, a variable step integrator may be used, in such a way to ensure not only the accuracy of the states, but also of the state derivatives.

The lap time optimisation program is based on a transient vehicle model. Obviously, achieving maximum efficiency for the vehicle dynamic simulation is the ideal scenario for subsequently applying optimisation technology. Automatic equation and code generation is the state of the art technology in the field of multibody dynamics. Tools such as AutoSim[1] allow the modeller to concentrate on the actual layout of the vehicle dynamics model in terms of bodies, degrees of freedom, forces,

[1] For more information see: `www.trucksim.com`.

etc., while they perform all the symbolic work to generate the dynamic equations in terms of first order ordinary differential equations and a complete, highly optimised simulation program. The current vehicle model implemented in FastLap is still relatively simple to benefit a great deal from this technology. Future upgrades to the vehicle model library in FastLap certainly will obtain such benefit.

Certain subsystems for the vehicle model, such as the tyre model, still needs to be coded separately and the efficiency depends on the ability of the programmer. On the other hand, when computational speed is an issue, the modeller can choose to simplify the representation of the vehicle model, provided that this does not spoil the relevance of the analysis to be performed. The tyre model in particular is one of the key issues, as the complete Magic Formula tyre model (Pacejka and Besselink, 1997) itself involves a significant proportion of the algebraic operations required to evaluate the vehicle model equations.(Automatic Differentiation allows to count the algebraic operations needed for a function evaluation, and for the seven degrees of freedom vehicle simulation nearly 40 % of the algebraic operations involved in computing the equations of motion is taken by the Magic Formula. On the one hand, when the aim is to study the limit behaviour of a vehicle, an accurate representation of the tyre non linear behaviour up to the saturation of the forces is required. However, the need for computational efficiency suggests to consider simpler mathematical representations for the tyre forces, like a neural network tyre model, or look-up tables. In the latter case a great deal of data would be needed to consider the whole range of variation of slip quantities, tyre load and camber angles. However, the evaluation would be significantly faster.

Another area to consider to further improve efficiency is obviously trying to reduce the number of iterations needed to achieve the desired accuracy. Improving the accuracy of the objective and constraint function evaluations and of their derivatives goes potentially in that direction, even though there is a trade off since the actual time to compute such quantities increases. One area which has not yet been considered is the interpolation method for the vehicle controls. The current linear interpolation scheme was chosen for simplicity, but it is now thought to be insufficiently accurate. From a numerical point of view, there is actually a mismatch between using a first order interpolation scheme for the control when the vehicle simulation is performed using a second order integration algorithm, and the optimisation using a super-linear algorithm. Furthermore, the noise which is observed particularly for the steer angle control at the solution, even though we are able to justify such noise to some extent, is thought to badly affect the convergence rate of the optimisation algorithm when near the solution. A smooth interpolation scheme, such as cubic spline interpolation, is expected to improve accuracy and efficiency. Limiting the rate of variation of the steer angle may be considered as well. As a final remark, though, we may consider that after all allowing a maximum of 200 iterations is a rather low limit for a large scale problem such as a full lap time optimisation. The default value for SNOPT would be the maximum between 1000 and the number of non-linear constraints, which is usually more than 2000 in a full lap time optimisation.

Various study cases have been successfully solved and the present accuracy in evaluating the minimum lap time proved to be sufficient to perform vehicle design and set-up parameters sensitivity analysis. A certain variability in the results, though has been observed. Imposing tighter tolerances seems to partially solve this problems, as it narrows the range in which the computed minimum lap time may fall. However, it appears to be in the nature of the problem to have several local minima close to each

other. As actually happens for real drivers, it is often possible to apply different driving strategies when there is a sequence of corners where the speed in one corner is affected by the speed from the previous one, and achieve nearly similar lap times. A possible solution to the problem would be to apply some sort of simulated annealing. For example, the optimisation would have to be stopped after a pre-determined number of iterations and to the current solution would be applied some noise. Such procedure has to be repeated a few times, always reducing the magnitude of the noise added, until convergence is achieved. Such procedure is bound to require more major iterations in total, but the chances to obtain a more global solution with one single run would certainly be increased.

Nonetheless, the quantitative predictions of the lap time sensitivity to parameter changes are very accurate and correlate very well with the practical findings on real Formula One race cars. The potential of the lap time optimisation method, though, is greater than simply evaluating the lap time. First of all, the nature of the optimal solution may be correlated with the handling qualities and the controllability of the vehicle on the limit. For example, some of the results seem to indicate that the worst case scenario for vehicle controllability issues is when both front and rear axles reach the lateral limit at the same time. In this situation the vehicle behaviour is very unpredictable and may quickly change from understeer to oversteer, a situation which would prevent a driver from achieving the maximum car performance. On the other hand, using the maximum friction capability of all the four tyres in cornering manoeuvres is one of the aims to maximise performance. The possibility offered by FastLap to investigate the limit transient behaviour of a racing car may be exploited also to optimise the handling in the typical operating conditions of a racing car.

Having achieved the possibility to simulate the minimum lap time for a certain car configuration, the next step is to optimise some of the vehicle design parameters to achieve the best possible performance on a datum circuit. The task could be achieved in two ways. One is to simply include the vehicle design parameters alongside the state and control optimisation parameters and let the optimisation algorithm vary them in order to minimise the objective function and meet the constraints. This is very straightforward, even though scaling with regard to the sensitivity of the objective with respect to the added parameters could be an issue. A second possibility is to build a secondary optimisation loop on top of the primary lap time optimisation loop. Firstly, the minimum time would be computed for the default vehicle configuration. Next, the sensitivity of the minimum time and of the constraints with respect to the chosen design parameters could be computed using Automatic Differentiation, and an improved set of parameters could be computed. The lap time optimisation would then have to be repeated, but since the control would already be very near to the optimum, it should only take a few iterations. The secondary loop would be repeated until the chosen design parameters converge. These methods, though, are only possible for scalar parameters. Vehicle configurations described by tables of data, such as aerodynamic maps, cannot be optimised in this way. In this case, the only option at the moment appears to be repeating the lap time evaluation for each anticipated vehicle configuration.

Finally, another scope of application for FastLap is the optimisation of vehicle control inputs other than steer angle and throttle/brake controls. Current Formula One technical regulations prevent most electronic driver aids. However, if for example a variable front and rear brake torque distribution was allowed, the question which would

arise would be what is the actual history of the brake torque distribution control over a certain manoeuvre which maximises the performance? Optimal control is the formal method to answer such questions.

List of publications

R. S. Sharp, D. Casanova and P. Symonds. (2000). 'A mathematical model for car steering, with design, tuning and performance results', Vehicle System Dynamics, 33(5), 289-326.

D. Casanova, R. S. Sharp, M. Final, B. Christianson and P. Symonds. 'Application of Automatic Differentiation to Race Car Performance Optimisation'. In: *Proceedings of Automatic Differentiation 2000: From Simulation to Optimisation*. Springer-Verlag, New York. In printing.

D. Casanova, R. S. Sharp and P. Symonds. (2000). 'Minimum Time Manoeuvring: The Significance of Yaw Inertia', Vehicle System Dynamics, 34(2), 2000, 77-115.

D. Casanova, R. S. Sharp and P. Symonds. (2000). 'On Minimum Time Optimisation of Formula One Cars: the Influence of Vehicle Mass'. In: Proceedings of AVEC'2000, 5th International Symposium on Advanced Vehicle Control , 22nd - 24th August 2000, Ann Arbor, Michigan, USA.

D. Casanova, R. S. Sharp and P. Symonds. 'Sensitivity to mass variations of the fastest possible lap of a Formula One car', Vehicle System Dynamics, in refereeing.

References

Allen, J. (1997). Computer Optimisation of cornering Line. Cranfield University, School of Mechanical Engineering, MSc thesis.

Anon. (1996). *MATLAB® Signal Processing Toolbox User's Guide*, version 4. The Mathworks Inc.

Ascher, U., Christiansed, J., Russell, R. D. (1981). 'Collocation Software for Boundary-Value ODEs'. ACM Transaction on Mathematical Software, vol. 7, no. 2, 209 - 222.

Bakker, E., Nyborg, L., Pacejka, H. B. (1987). Tyre modelling for use in vehicle dynamics studies. SAE 870421.

Bartholomew-Biggs, M.C., Bartholomew-Biggs, L., Christianson, D. B. (1994). 'Optimisation and Automatic Differentiation in ADA: Some Practical Experience'. Optimisation Methods and Software, no. 4, 47-73.

Betts, T. J., Huffman, W. P. (1991). 'Trajectory Optimisation on a Parallel Processor'. Journal of Guidance, vol. 14, no. 2.

Betts, T. J. (1993). Trajectory Optimisation using Sparse Sequential Quadratic Programming. In: *Computational Optimal Control*, International Series of Numerical Mathematics, vol. 111, Birkhauser, Basel.

Betts, T. J. (1994). Issues in the direct transcription of optimal control problems to sparse non linear programs. In: *Computational Optimal Control*, International Series of Numerical Mathematics, vol. 115, Birkhauser, Basel.

Bryson, A.E., Ho, Y. (1975). *Applied Optimal Control.* Hemisphere Publishing Corporation, USA.

Christianson, D. B. (1992). 'Automatic Hessian by Reverse Accumulation'. I.M.A. Journal of Numerical Analysis, vol. 12, 135-150.

Cloutier, J.R., Mohanty, B.P., Miele, A. (1977). Sequential Conjugate Gradient Restoration Algorithm for Optimal Control Problems with non Differential Constraints - Part 1 and 2. Aero - Astronautics Reports, no. 126 and 127. Rice University, Houston, Texas.

Cossalter, V., Da Lio, M., Lot, R., Fabbri, L. (1999). 'A New General Method for the Evaluation of Vehicle Manoeuvrability with Special Emphasis on Motorcycles'. Vehicle System Dynamics, vol. 31, 113 - 135.

Dixon, L. C. W., Price, R. C. (1989). 'Truncated Newton Method for Sparse Unconstrained Optimisation Using Automatic Differentiation'. Journal of Optimisation Theory and Applications, vol. 60, no. 2.

Dixon, J. C. (1996). *Tires, suspensions and handling.* SAE, Warrendale.

Eberhard, P., Bischof, C. (1999). 'Automatic Differentiation of Numerical Differentiation Algorithms'. Mathematics of Computation, vol. 68, no. 226, 717 – 731.

Enright, P. J., Conway, B. A. (1990). Discrete approximation to optimal trajectories using direct transcription and non-linear programming. In: *AIAA/AHS Astrodynamics Conference*, Part 2, 20th – 22nd August, Portland, Oregon.

Fletcher, R. (1987). *Practical Methods of Optimisation.* Wiley – Interscience, United Kingdom.

Fujiuka, T., Kimura, T. (1992). 'Numerical Simulation of Minimum time Cornering Behaviour'. JSAE Review, vol. 13, no. 1, 44 - 51.

Gadola, M., Vetturi, D., Cambiaghi, D., Manzo, L. (1996). A Tool for Lap Time Simulation. In: Proceedings of the SAE Motorsport Engineering Conference & Exposition, Dearborn, Michigan, USA.

Gill, P. E., Murray, W., Wright, M.H. (1981). *Practical Optimisation.* Academic Press Inc. (London) Ltd.

Gill, P. E., Murray, W., Saunders, M. A. (1994). Large-scale SQP methods and their application in trajectory optimisation. In: *Computational Optimal Control*, International Series of Numerical Mathematics, vol. 115, Birkhauser, Basel.

Gill, P. E., Murray, W., Saunders, M. A. (1997). SNOPT: An SQP algorithm for large-scale constrained optimisation. Numerical Analysis Report 97-2, Department of Mathematics, University of California, San Diego, La Jolla, CA.

Griewank, A. (1989). On Automatic Differentiation. In: *Mathematical Programming: Recent Development and Application*, edited by M. Iri and K. Tanabe, Kluwer Academic Publisher.

Griewank, A. (2000). *Evaluating Derivatives. Principle and Techniques of Algorithmic Differentiation.* SIAM, Society for Industrial and Applied Mathematics, Philadelphia, 2000.

Griffiths, R. (1992). Minimum Lap Time Simulation of a Racing Car. Cranfield Institute of Technology, School of Mechanical Engineering, MSc Thesis.

Guo, K., Guan, H. (1993). 'Modelling of Driver/Vehicle Directional Control System'. Vehicle System Dynamics, vol. 22, 141-184.

Hendrikx, J.P.M., Meijlink, T.J.J., Kriens, R.F.C. (1996). 'Application of Optimal Control Theory to Inverse Simulation of Car Handling'. Vehicle System Dynamics, vol. 26, 449 - 462.

Jennings, M.J. (1996). Dynamic Simulation of Race Car Performance. In: Proceedings of the SAE Motorsport Engineering Conference & Exposition, Dearborn, Michigan, USA.

Kernighan, B.W., Ritchie, D. M. (1988). *The C Programming Language.* Englewood Cliff, Prentice Hall, New Jersey, USA.

Kirk, D.E. (1970). *Optimal Control Theory - An Introduction.* Englewood Cliff, Prentice Hall, New Jersey, USA.

Kraft, D. (1994). 'Algorithm 733: TOMP - Fortran Modules for Optimal Control Calculations'. ACM Transaction on Mathematical Software, vol. 20, 262 - 281.

Kraft, D., Bulirsch, R. (1994). *Computational Optimal Control.* International Series of Numerical Mathematics, vol. 115, Birkhauser, Basel.

Kwakernaak, H., Sivan, R. (1972). *Linear Optimal Control Systems.* Wiley – Interscience, USA.

La Joie, J. C. (1994). Race car performance optimisation. SAE Technical Report 942492.

MacAdam, C. C. (1981). 'Application of an optimal preview control for simulation of closed-loop automobile driving'. IEEE Transactions on Systems, Man and Cybernetics, SMC-11(6), 393-399.

McAdam, C.C., Johnson, G.E. (1996). 'Application of Elementary Neural Networks and Preview Sensors for Representing Driver Steering Behaviour'. Vehicle System Dynamics, vol. 25, no. 1.

Mehlhorn, R., Dinkelmann, M., Sachs, G., Application of Automatic Differentiation to Optimal Control Problems. In: *Computational Optimal Control*, International Series of Numerical Mathematics, vol. 115, Birkhauser, Basel.

Metz, D., Williams, D. (1989). 'Near Time-optimal Control of Racing Vehicles'. Automatica, vol 25, no. 6. 841-857.

Milliken, W.F., Milliken, D.L. (1995). *Race Car Vehicle Dynamics.* SAE International, USA.

Moss, S., Pomeroy, L. (1963). *Design and Behaviour of the Racing Car.* William Kimber, London.

Pacejka, H.B., Besselink, I. J. M. (1997). 'Magic Formula Tyre Model with Transient Properties'. Vehicle System Dynamics, vol. 27, 234-249.

Pontryagin, L. S., Boltyanskii, V. G., Gamkrelidze, R. V., Mishchenko, E. F. (1962). *The Mathematical Theory of Optimal Processes.* Interscience Publishers, Inc., New York.

Press, W. H., Flemming, B. P., Vetterling, W. T., Teukolsky, S. A. (1992). *Numerical Recipes in C.* Cambridge University Press, Second Edition.

Sage, A.P. (1977). *Optimum Systems Control.* Englewood Cliff, Prentice Hall, New Jersey.

Schildt, H. (1998). *C++ from the Ground Up.* Osborne/McGraw-Hill. Brekeley, California, Second Edition.

Seywald, H., Kumar, H. H., Wetzel, T. A. (1996). Does the principle of optimality hold along a collocation solution? In: *AIAA Guidance Navigation and Control Conference.* 29th – 31st July, San Diego, CA, USA.

Sharp, R.S. (1994). 'The application of multibody computer codes to road vehicle dynamics modelling problems'. Proc. IMechE, Vol. 208.

Sharp, R. S. (1995). 'Preview control of active suspensions'. In: *Smart Vehicles*, J. P. Pauwelussen and H. B. Pacejka (eds), Swets and Zeitlinger, 166-182.

Sharp, R.S., Prokop, G. (1995). 'Performance Enhancement of limited bandwidth active automotive suspensions by road preview'. Proc. IEE, Control Theory application, Vol. 142, No. 2.

Sharp, R.S. (1996). Vehicle Dynamics Modelling with the aid of a Symbolic Multibody Code. In: Proceedings of AVEC'96, vol. 2, pages 971-984.

Stroustrup, B. (1997). *The C++ Programming Language.* AT&T Labs, Murray Hill, New Jersey, Third Edition.

Stryk, von O. (1993) Numerical Solution of Optima Control Problems by Direct collocation. In: *Computational Optimal Control*, International Series of Numerical Mathematics, vol. 111, Birkhauser, Basel.

Stryk, von O., Pesch, H. J., Nerz, E., Bulirsh, R. (1993). Combining direct and indirect methods in optimal control: range optimisation of a Hang Glider. In: *Computational Optimal Control*, International Series of Numerical Mathematics, Vol 111, Birkhauser, Basel.

Tarassenko, L. (1998). *A guide to neural computing applications*, Arnold, London (co-published by John Wiley for the Americas).

Valtetsiotis, V. (1999). *A Discrete-Time Optimal Preview Control Driver Model.* Cranfield University, School of Mechanical Engineering, MSc thesis.

Weir, D. H. and McRuer, D. T. (1968). 'A theory for driver steering control of motor vehicles', Highway Research Record, 247, 7-39.

Wu, K.A., Miele, A. (1978a). Sequential Conjugate Gradient Restoration Algorithm for Optimal Control Problems with General Boundary Conditions. Aero - Astronautics Reports, No 144. Rice University, Houston, Texas.

Wu, K.A., Miele, A. (1978b). Sequential Conjugate Gradient Restoration Algorithm for Optimal Control Problems with non Differential Constraints and General Boundary Conditions. Aero - Astronautics Reports, No 145. Rice University, Houston, Texas.

Yeo, C. L. (1998). *Influence of Controlled Transmission Elements on Vehicle Behaviour.* Cranfield Institute of Technology, School of Mechanical Engineering, MSc thesis.

Appendix A.

Vehicle Model Equations

Referring to figure A.1, the equations of motion of the seven degrees of freedom vehicle model read (for the meaning of the symbols see the Notation section):

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \left[(F_{x1} - F_{x2}) \cdot \frac{t_f}{2} + (F_{x3} - F_{x4}) \cdot \frac{t_r}{2} + (F_{y1} + F_{y2}) \cdot l_f - (F_{y3} + F_{y4}) \cdot l_r \right] \frac{1}{I_{zz}}$$

$$\dot{x}_3 = [F_{x1} + F_{x2} + F_{x3} + F_{x4} - F_{ax}] \frac{1}{M} + x_2 x_4$$

$$\dot{x}_4 = [F_{y1} + F_{y2} + F_{y3} + F_{y4}] \frac{1}{M} - x_2 x_3$$

$$\dot{x}_5 = x_3 \cdot \cos(x_1) - x_4 \cdot \sin(x_1)$$

$$\dot{x}_6 = x_3 \cdot \sin(x_1) + x_4 \cdot \cos(x_1)$$

$$\dot{x}_7 = x_8$$

$$\dot{x}_8 = (T_1 - F_{xw1} \cdot R_f) \frac{1}{J_{wf}}$$

$$\dot{x}_9 = x_{10}$$

$$\dot{x}_{10} = (T_2 - F_{xw2} \cdot R_f) \frac{1}{J_{wf}}$$

$$\dot{x}_{11} = x_{12}$$

$$\dot{x}_{12} = \frac{(T_3 - F_{x3} \cdot R_r) \cdot \left(J_{wr} + J_m \cdot \left(\frac{G_R}{2} \right)^2 \right) - (T_4 - F_{x4} \cdot R_r) \cdot J_m \cdot \left(\frac{G_R}{2} \right)^2}{J_{wr}^2 + 2 \cdot J_{wr} \cdot J_m \cdot \left(\frac{G_R}{2} \right)^2} \qquad \text{Eq. A.1}$$

$$\dot{x}_{13} = x_{14}$$

$$\dot{x}_{14} = \frac{(T_4 - F_{x4} \cdot R_r) \cdot \left(J_{wr} + J_m \cdot \left(\frac{G_R}{2} \right)^2 \right) - (T_3 - F_{x3} \cdot R_r) \cdot J_m \cdot \left(\frac{G_R}{2} \right)^2}{J_{wr}^2 + 2 \cdot J_{wr} \cdot J_m \cdot \left(\frac{G_R}{2} \right)^2}$$

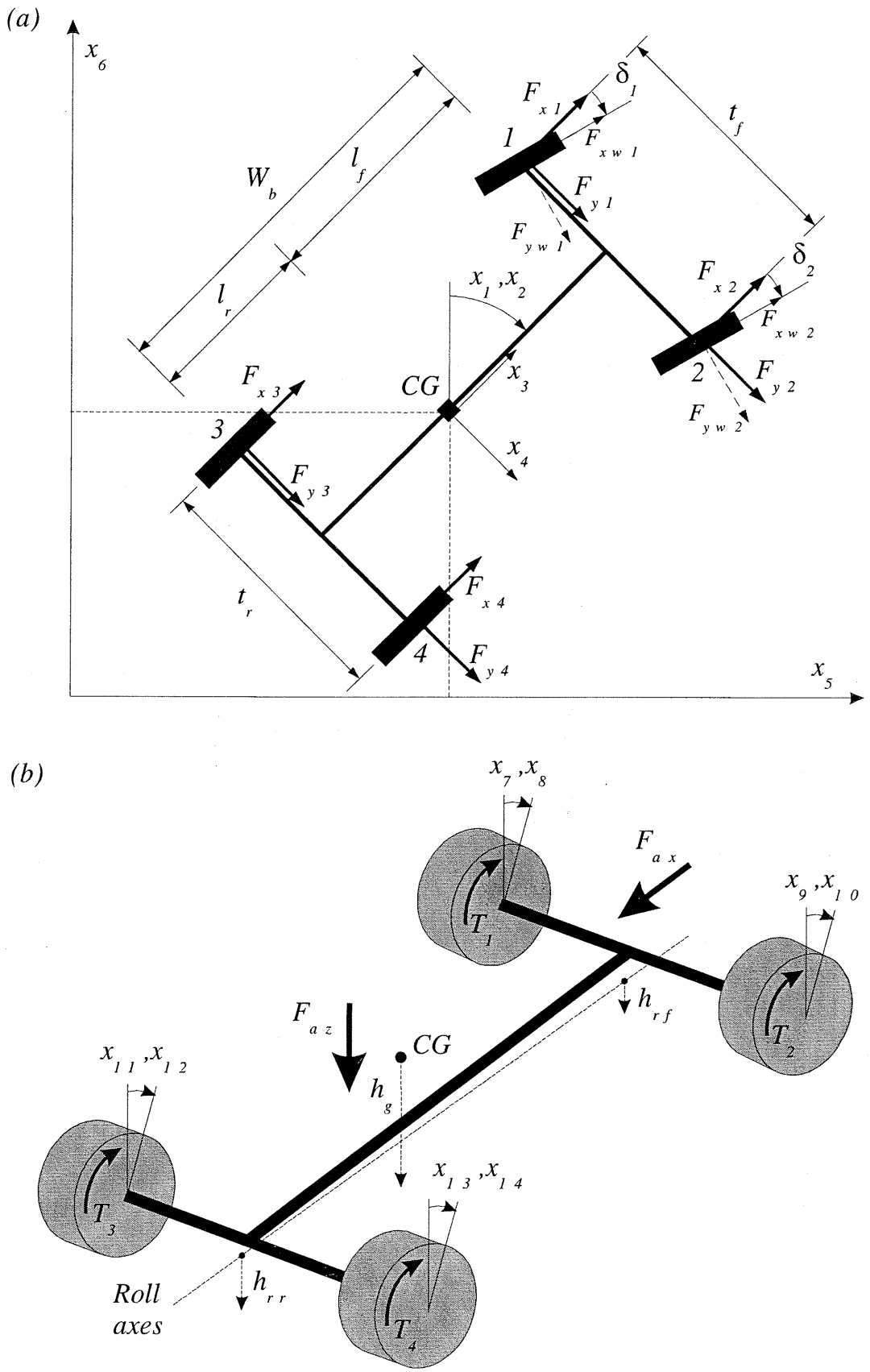


Figure A.1 Scheme of the vehicle model.

For the front wheels, the tyre forces are firstly evaluated into the wheel's reference axis system (F_{xw1}, F_{yw1}, F_{xw2} and F_{yw2} in figure A.1 (a)) and then projected onto the longitudinal and lateral directions:

$$\begin{cases} F_{x1} = F_{xw1} \cdot \cos(\delta_1) - F_{yw1} \cdot \sin(\delta_1) \\ F_{y1} = F_{xw1} \cdot \sin(\delta_1) + F_{yw1} \cdot \cos(\delta_1) \\ F_{x2} = F_{xw2} \cdot \cos(\delta_2) - F_{yw2} \cdot \sin(\delta_2) \\ F_{y2} = F_{xw2} \cdot \sin(\delta_2) + F_{yw2} \cdot s\cos(\delta_2) \end{cases} \quad \text{Eq. A.2}$$

The individual steer angles for the front wheels are assumed to be equal:

$$\delta_1 = \delta_2 = \delta \quad \text{Eq. A.3}$$

The model of the powertrain includes:

- Engine output torque described using an experimental engine map;
- Global transmission ratios;
- Torque sensitive, limited slip differential.

The experimental engine map allows to evaluate the engine output torque as a function of two parameters, the throttle aperture and the engine rotational velocity. At any time in the simulation, the engine rotational velocity may be evaluated from the rear wheel velocities by applying the differential rule:

$$\omega_E = \frac{(x_{12} + x_{14})}{2} \cdot G_R \cdot \frac{30}{\pi} [rpm] \quad \text{Eq. A.4}$$

Here, the gear ratio G_R is selected upon the vehicle forward velocity x_3. Using the current values of the longitudinal control variable u_{tb} and the engine rotational velocity, the engine output torque T_E is estimated from the engine map $\mathbf{E_{trq}}$ by means of bi-linear interpolation:

$$T_E = \text{bi\_linear\_interp}(\mathbf{E_{rpm}}, \mathbf{E_{th}}, \mathbf{E_{trq}}, \omega_E, u_{tb}) \quad \text{Eq. A.5}$$

The torque applied to each driven wheel may then be computed as follows:

$$\begin{cases} T_3 = \frac{T_E \cdot G_R}{2} + \Delta_T \\ T_4 = \frac{T_E \cdot G_R}{2} - \Delta_T \end{cases} \quad \text{Eq. A.6}$$

Δ_T is the amount of torque transfer due to the limited slip differential. The simple model used to compute the torque transfer is based on the Salisbury type differential (Milliken and Milliken, 1995). In this differential clutch packs are used to progressively lock the

two output shafts together. The axles of the differential pinions rest against wedges which are spread by the input torque, so that they apply a pressure to the clutch packs which is proportional to the input torque. Furthermore, the wedges may have different angles to give different characteristics in driving and in overrun. The amount of torque transfer may then be evaluated as a constant portion of the input torque. Two different torque gains G_{D2_D} and G_{D2_O} are used to represent the different characteristics of the differential in driving and in overrun respectively. Also a constant off-set G_{D1} is included to account for the natural friction in the differential. According to the SAE sign conventions, the driving torque is negative and the braking torque is positive. Hence, if T_D is the input torque to the differential cage, the torque transfer reads:

$$\begin{aligned} T_T &= -G_{D1} + G_{D2\_D} \cdot T_D \quad \text{if} \quad T_D < 0 \quad (\textit{driving}) \\ T_T &= -G_{D1} - G_{D2\_O} \cdot T_D \quad \text{if} \quad T_D > 0 \quad (\textit{overrun}) \end{aligned} \qquad \text{Eq. A.7}$$

Eq. A.7 always returns a negative value for the torque transfer. The direction of the torque transfer which is actually applied to the rear wheels depends on the sign of their differential velocity:

$$\Delta T = T_T \cdot \text{sign}(\omega_{RR} - \omega_{LR}) \qquad \text{Eq. A.8}$$

When the equations of motion for the rear wheels are derived, it is necessary to take into account that the torque values T_3 and T_4 do not only accelerate the rotating masses associated with the wheels. The inertia due to the rotating mass of the engine has to be taken into account, since it depends on the gear ratio and its influence is significant in lower gears. Let J_m be the inertia of the rotating mass of the engine. The input torque to the differential cage T_D may be written as follows:

$$T_D = T_E \cdot G_R - J_m \cdot \dot{\omega}_E \cdot G_R = T_E \cdot G_R - J_m \cdot \frac{(\dot{x}_{12} + \dot{x}_{14})}{2} \cdot G_R^2 \qquad \text{Eq. A.9}$$

Neglecting the inertia of the differential, we may write that the torque T_D is divided between the two half shafts accounting for the torque transfer Δ_T as follows:

$$\begin{cases} T_3' = \dfrac{T_D}{2} + \Delta_T \\ T_4' = \dfrac{T_D}{2} - \Delta_T \end{cases} \qquad \text{Eq. A.10}$$

Since T_3' and T_4' are the actual torque values applied to the rear wheels, we may write the following equations for the rear wheel spin dynamics:

$$\begin{cases} T_3' - J_{wr} \cdot \dot{x}_{12} - F_{x3} \cdot R_r = 0 \\ T_4' - J_{wr} \cdot \dot{x}_{14} - F_{x4} \cdot R_r = 0 \end{cases} \qquad \text{Eq. A.11}$$

Eq. A.9 may be modified using Eq. A.10 as follows:

$$T_3' + T_4' = T_E \cdot G_R - J_m \cdot \frac{(\dot{x}_{12} + \dot{x}_{14})}{2} \cdot G_R^2 \qquad \text{Eq. A.12}$$

From Eq. A.10 we may also derive:

$$T_3' = T_4' + 2 \cdot \Delta_T \qquad \text{Eq. A.13}$$

Hence, by substituting Eq. A.13 into Eq. A.12 we obtain:

$$T_4' = \left(\frac{T_E \cdot G_R}{2} - \Delta_T \right) - J_m \cdot \frac{(\dot{x}_{12} + \dot{x}_{14})}{4} \cdot G_R^2 = T_4 - J_m \cdot \frac{(\dot{x}_{12} + \dot{x}_{14})}{4} \cdot G_R^2 \qquad \text{Eq. A.14}$$

In the same way we obtain for the rear left wheel:

$$T_3' = T_3 - J_m \cdot \frac{(\dot{x}_{12} + \dot{x}_{14})}{4} \cdot G_R^2 \qquad \text{Eq. A.15}$$

Substituting Eqs. A.14 and A.15 into Eq. A.11, and solving for $\dot{x}_{12}$ and $\dot{x}_{14}$, allows to obtain the equations of motion for the rear wheels as functions of the torque values T_3 and T_4 computed by using Eq. A.6.

When braking, the amount of negative torque applied to the four wheels is evaluated by multiplying the maximum braking torque available by the value of the longitudinal control variable:

$$T_B = u_{tb} \cdot T_{B\max} \qquad u_{tb} < 0 \qquad \text{Eq. A.16}$$

T_B is shared between the front and rear axles according to the constant coefficients B_f and B_r. Then, the braking torque is divided evenly between the wheels on the same axle. The braking effect of the engine is accounted for by evaluating Eq. A.5 with u_{tb} equal to 0. Also the torque transfer due to the differential is included. Hence, the expressions of the braking torque for each wheel read:

$$\begin{cases} T_1 = \dfrac{T_B \cdot B_f}{2} \\ T_2 = \dfrac{T_B \cdot B_f}{2} \\ T_3 = \dfrac{T_B \cdot B_r}{2} + \dfrac{T_E \cdot G_R}{2} + \Delta_T \\ T_4 = \dfrac{T_B \cdot B_r}{2} + \dfrac{T_E \cdot G_R}{2} - \Delta_T \end{cases} \qquad \text{Eq. A.17}$$

The aerodynamic drag and down-force in Eq. A.1 are evaluated as follows:

$$F_{ax} = \frac{1}{2}\rho \cdot S_f \cdot C_x \cdot x_3^2$$
$$F_{az} = \frac{1}{2}\rho \cdot S_f \cdot C_z \cdot x_3^2 \qquad \text{Eq. A.18}$$

The vertical load acting on each wheel is evaluated including a steady state approximation of the lateral and longitudinal load transfers. Firstly, the static distribution of the vehicle weight and the aerodynamic down-force on the front and rear axles is calculated as follows:

$$F_{fz\_static} = \left(M \cdot g \cdot l_r + F_{az} \cdot D_f\right) / W_b$$
$$F_{rz\_static} = \left(M \cdot g \cdot l_f + F_{az} \cdot D_r\right) / W_b \qquad \text{Eq. A.19}$$

Then, the longitudinal load transfer, which occurs in driving or braking conditions, is evaluated by considering the equilibrium of the vehicle about its centre of gravity. Assuming the aerodynamic drag acts at the height of the CG, the expression for the longitudinal load transfer reads:

$$\Delta F_{z\_long} = \left(F_{x1} + F_{x2} + F_{x3} + F_{x4}\right)\frac{h_g}{W_b} \cong \left(\frac{T_1 + T_2}{R_f} + \frac{T_3 + T_4}{R_r}\right)\frac{h_g}{W_b} \qquad \text{Eq. A.20}$$

Since the longitudinal tyre forces are functions of the tyre vertical loads and therefore unknown, their values are approximated by substituting the torque applied to each wheel divided by the wheel radius. In doing this, the contribution of the inertia of the rotating masses is neglected and the longitudinal load transfer will be slightly overestimated.

The lateral load transfer is estimated by evaluating the steady state cornering equilibrium of the vehicle. It is assumed that the roll axis has a fixed position and that the roll stiffness distribution is constant. The vehicle chassis is considered infinitely stiff, hence front and rear roll angles are the same. Furthermore, the contribution of the unsprung masses is neglected. Referring to figure A.2, which represents the front end of the vehicle, the roll equilibrium of the body about the roll axis reads:

$$T_{rf} - M_f \cdot \frac{x_3^2}{r_t} \cdot \left(h_g - h_{rf}\right) \cdot \cos(\phi) + M_f \cdot g \cdot \left(h_g - h_{rf}\right) \cdot \sin(\phi) - C_{f\phi} \cdot \phi = 0 \qquad \text{Eq. A.21}$$

Analogously, the roll equilibrium of the whole vehicle front end about a ground origin located in the middle of the vehicle reads:

$$\left(\frac{M_f \cdot g}{2}+\Delta F_{fz}\right)\cdot\frac{t_f}{2}-\left(\frac{M_f \cdot g}{2}-\Delta F_{fz}\right)\cdot\frac{t_f}{2}+$$
$$-M_f \cdot\frac{x_3^2}{r_t}\cdot\left[h_{rf}+\left(h_g-h_{rf}\right)\cdot\cos(\phi)\right]+M_f \cdot g\cdot\left(h_g-h_{rf}\right)\cdot\sin(\phi)+T_{rf}=0 \qquad \text{Eq. A.22}$$

Eq. A.22 may be simplified by assuming small roll angles, i.e. $\cos(\phi)\cong 1$ and $\sin(\phi)\cong\phi$. Furthermore, solving Eq. A.21 for T_{rf} and substituting in Eq. A.22 yields:

$$\Delta F_{fz}\cdot t_f - M_f\cdot\frac{x_3^2}{r_t}\cdot h_{fr}+C_{f\phi}\cdot\phi=0 \qquad \text{Eq. A.23}$$

From Eq. A.23, the front lateral load transfer may be derived:

$$\Delta F_{fz}=\frac{M_f\cdot\frac{x_3^2}{r_t}\cdot h_{fr}-C_{f\phi}\cdot\phi}{t_f} \qquad \text{Eq. A.24}$$

If we now consider the roll equilibrium of the whole vehicle body, the roll angle resulting from a constant lateral acceleration of the centre of mass may be evaluated as follows:

$$\phi=-\frac{E\cdot\frac{x_3^2}{r_t}}{C_{f\phi}+C_{r\phi}-E\cdot g} \qquad \text{Eq. A.25}$$

where the term E reads:

$$E=M_f\cdot\left(h_g-h_{rf}\right)+M_r\cdot\left(h_g-h_{rr}\right) \qquad \text{Eq. A.26}$$

Eq. A.26 may be simplified by using the longitudinal position of the centre of mass to compute the front and rear load distribution:

$$\begin{aligned}E&=M\cdot\left[\frac{l_r}{W_b}\cdot\left(h_g-h_{rf}\right)+\frac{l_f}{W_b}\cdot\left(h_g-h_{rr}\right)\right]=\\&=M\cdot\left(h_g-\frac{l_r}{W_b}\cdot h_{rf}-\frac{l_f}{W_b}\cdot h_{rr}\right)=\\&=M\cdot\left(h_g-h_{rc}\right)\end{aligned} \qquad \text{Eq. A.27}$$

Here, h_{rc} is the roll centre height at the position of the vehicle centre of gravity. Next, Eq. A.27 may be substituted in Eq. A.25. The resulting expression may be simplified further by observing that in general $C_{f\phi}+C_{r\phi}>>M\cdot g\cdot\left(h_g-h_{rc}\right)$:

$$\phi = -\frac{M \cdot (h_g - h_{rc}) \cdot \frac{x_3^2}{r_t}}{C_{f\phi} + C_{r\phi} - M \cdot g \cdot (h_g - h_{rc})} \cong -\frac{M \cdot (h_g - h_{rc}) \cdot \frac{x_3^2}{r_t}}{C_{f\phi} + C_{r\phi}} \qquad \text{Eq. A.28}$$

Substituting Eq. A.28 into Eq. A.24 yields the expression for the lateral load transfer for the front axle as function of the vehicle states and parameters. However, since it is not normally possible to compute the actual turning radius r_t, it is more convenient to use the following expression to compute the vehicle lateral acceleration:

$$\frac{x_3^2}{r_t} = x_2 \cdot x_3 \qquad \text{Eq. A.29}$$

We are now able to write the expressions of the lateral load transfer for the front and rear axles read respectively:

$$\Delta F_{fz\_lat} = \frac{x_2 \cdot x_3 \cdot M}{t_f}\left[\left(l_r \cdot h_{rf}\right) / W_b + R_{sf}\left(h_g - h_{rc}\right)\right]$$
$$\Delta F_{rz\_lat} = \frac{x_2 \cdot x_3 \cdot M}{t_r}\left[\left(l_f \cdot h_{rr}\right) / W_b + R_{sr}\left(h_g - h_{rc}\right)\right] \qquad \text{Eq. A.30}$$

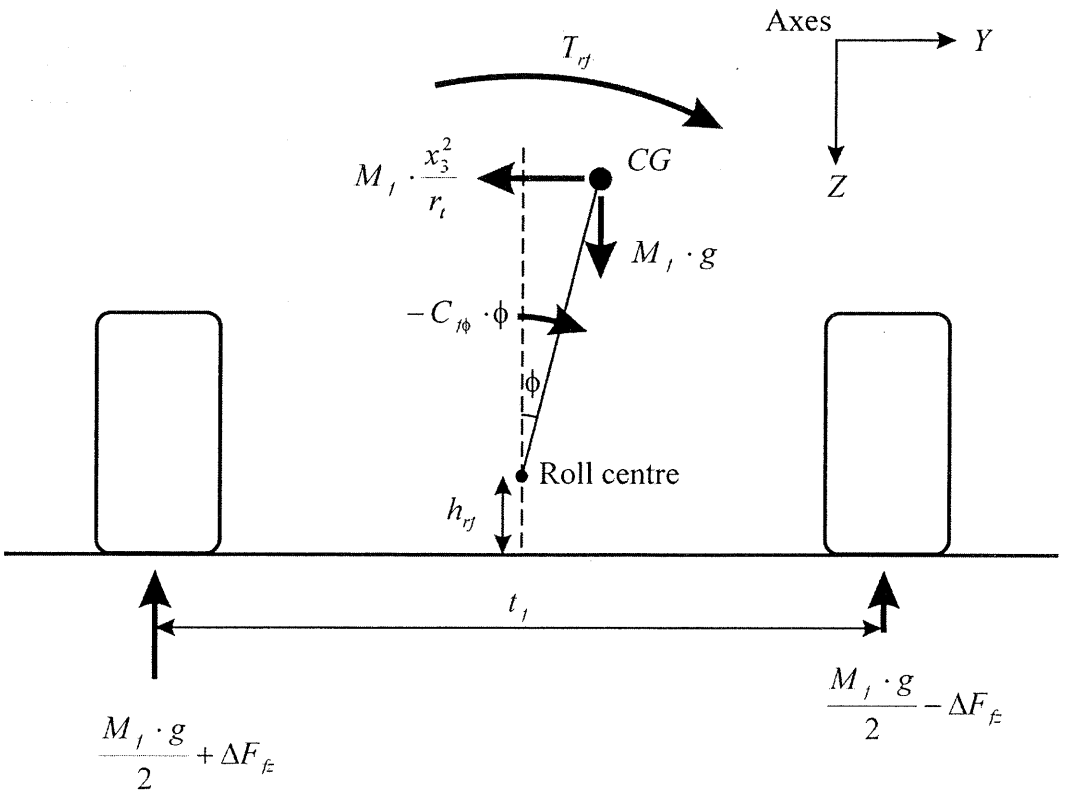


Figure A.2 Schematic rear view of the front suspension.

Finally, the above expressions may be combined to evaluate the vertical load on each wheel. According to the SAE conventions, which implies negative tyre loads, we may write:

$$\begin{aligned} F_{z1} &= -\left[\frac{1}{2}F_{fz\_static} - \frac{1}{2}\Delta F_{z\_long} + \Delta F_{fz\_lat}\right] \\ F_{z2} &= -\left[\frac{1}{2}F_{fz\_static} - \frac{1}{2}\Delta F_{z\_long} - \Delta F_{fz\_lat}\right] \\ F_{z3} &= -\left[\frac{1}{2}F_{rz\_static} + \frac{1}{2}\Delta F_{z\_long} + \Delta F_{rz\_lat}\right] \\ F_{z4} &= -\left[\frac{1}{2}F_{rz\_static} + \frac{1}{2}\Delta F_{z\_long} - \Delta F_{rz\_lat}\right] \end{aligned} \qquad \text{Eq. A.31}$$

The tyre forces are evaluated by using the Magic Formula Tyre Model which features the use of weighting functions to represent the tyre behaviour at combined slip conditions (Pacejka and Besselink, 1997). We may formally write:

$$\begin{cases} F_{ywi(combined)} = F_{ywi(free\ rolling)}(\alpha_i, \gamma_i, F_{zi}) \cdot G_y(\alpha_i, k_i, \gamma_i, F_{zi}) \\ F_{xwi(combined)} = F_{xwi(pure\ slip)}(k_i, F_{zi}) \cdot G_x(\alpha_i, k_i) \end{cases} \quad i = 1,2,3,4 \qquad \text{Eq. A.32}$$

The tyre forces are referred to the wheel reference axes system. For the steered wheels, the longitudinal and lateral forces are projected onto the vehicle reference axes system using equations A.2. A value for the static camber γ_i is assigned to each wheel. The tyre slip quantities are evaluated with the assumption of small angles and they are referred to the centre of the contact area of each wheel. The expressions for the slip angles in degrees and the longitudinal slip in percent read respectively:

$$\begin{cases} \alpha_1 = -\delta_1 + \dfrac{x_4 + l_f x_2}{x_3 + x_2 \cdot \dfrac{t_f}{2}} \dfrac{180}{\pi}; \quad \alpha_2 = -\delta_2 + \dfrac{x_4 + l_f x_2}{x_3 - x_2 \cdot \dfrac{t_f}{2}} \dfrac{180}{\pi} \\ \alpha_3 = \dfrac{x_4 - l_r x_2}{x_3 + x_2 \cdot \dfrac{t_r}{2}} \dfrac{180}{\pi}; \quad \alpha_4 = \dfrac{x_4 - l_r x_2}{x_3 - x_2 \cdot \dfrac{t_r}{2}} \dfrac{180}{\pi} \end{cases} \qquad \text{Eq. A.33}$$

$$\begin{cases} k_1 = -\left(1 - \dfrac{x_8}{x_3 + x_2 \cdot \dfrac{t_f}{2}} R_f\right) \times 100; \quad k_2 = -\left(1 - \dfrac{x_{10}}{x_3 - x_2 \cdot \dfrac{t_f}{2}} R_f\right) \times 100 \\ k_3 = -\left(1 - \dfrac{x_{12}}{x_3 + x_2 \cdot \dfrac{t_r}{2}} R_r\right) \times 100; \quad k_4 = -\left(1 - \dfrac{x_{14}}{x_3 - x_2 \cdot \dfrac{t_f}{2}} R_r\right) \times 100 \end{cases} \qquad \text{Eq. A.34}$$

Two sets of tyre data were employed. One is representative of the typical tyre forces for a Formula One car on dry tarmac, the other of the typical tyre forces for a rally car on a low friction surface, e.g. gravel or mud. Figures A.3 and A.4 show an example of such forces at pure slip for a range of loads representative of the vehicle working conditions.

When presenting results we often referred to the tyre saturation level as a measure for evaluating whether the vehicle is working more or less close to the limit of its performance envelope. We define the saturation level of a tyre either in longitudinal or lateral direction as follows. Let us consider a tyre generating the lateral force F_{yi} and the longitudinal force F_{xi} at the actual working conditions a_i, k_i, F_{zi} and γ_i. The lateral saturation is defined as the ratio between F_{yi} and the maximum lateral force obtainable while keeping k_i constant:

$$F_{y\_MAX} = \max_{\alpha}\left(F_{y(combined)}\right) \quad \text{constant longitudinal slip } k_i, \alpha \text{ varying}$$

$$Lat\_sat = \frac{F_{yi}}{F_{y\_MAX}} \times 100 \quad [\%] \qquad \text{Eq. A.35}$$

Similarly, for the longitudinal saturation we may write:

$$F_{x\_MAX} = \max_{k}\left(F_{x(combined)}\right) \quad \text{constant slip angle } \alpha_i, k \text{ varying}$$

$$Long\_sat = \frac{F_{xi}}{F_{x\_MAX}} \times 100 \quad [\%] \qquad \text{Eq. A.36}$$

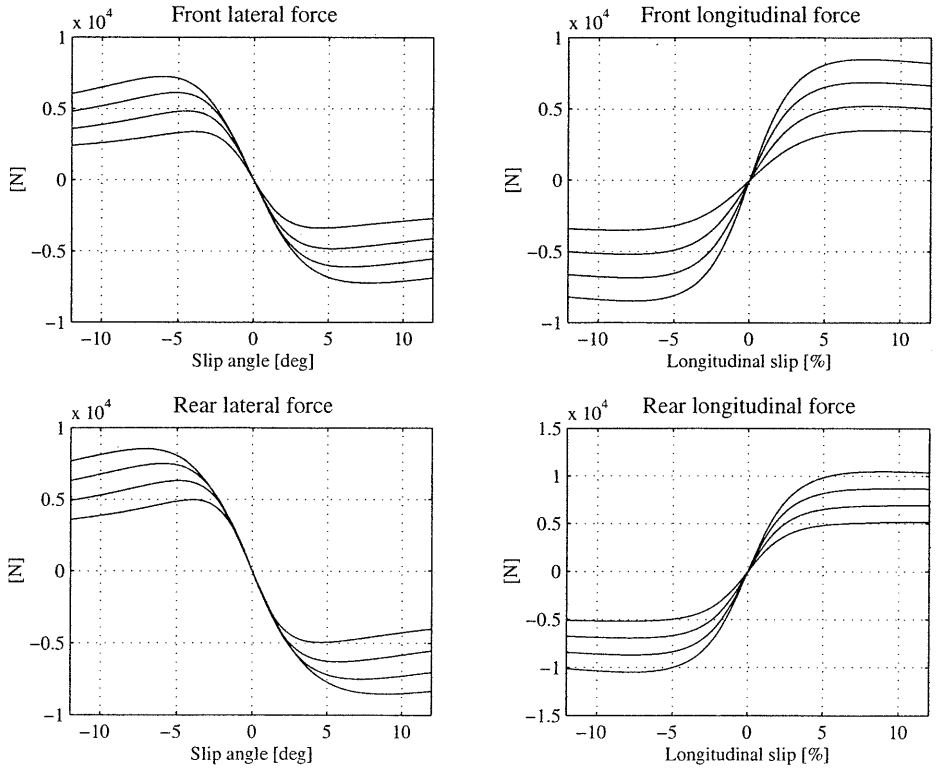


Figure A.3 Tyre force characteristics, F1 car, front loads equal to 2, 3, 4 and 5 kN, rear loads equal to 3, 4, 5 and 6 kN.

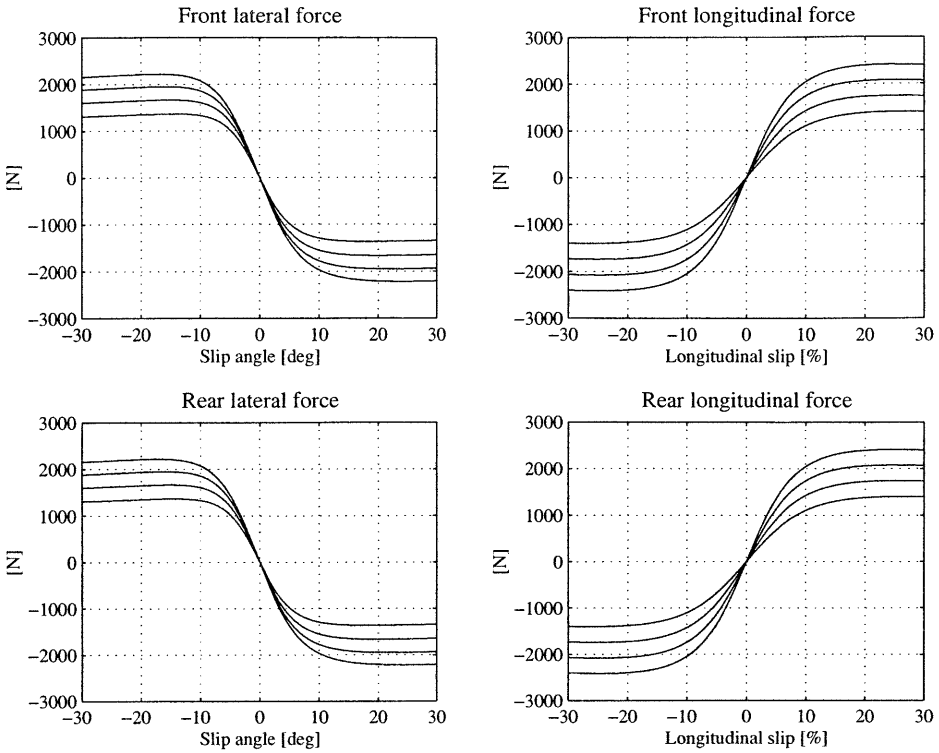


Figure A.4 Tyre force characteristics, rally car, front and rear loads equal to 2, 2.5, 3 and 3.5 kN.

Appendix B.

AutoSim Program List

The vehicle model described in Appendix A has been implemented in a C/C++ simulation program using the symbolic multibody code generator AutoSim, version 2.5. The AutoSim documentation may be found at `www.trucksim.com`. Further information on the use of the software may be found in (Sharp, 1994, Sharp, 1996).

The AutoSim vehicle model includes the following eight bodies:

- `main`: the vehicle chassis with three degrees of freedom, lateral longitudinal and yaw;
- `Wlf`: left front wheel, one rotational degree of freedom with respect to the chassis;
- `Wrf`: right front wheel, one rotational degree of freedom with respect to the chassis;
- `Wlr`: left rear wheel, one rotational degree of freedom with respect to the chassis;
- `Wrr`: right rear wheel, one rotational degree of freedom with respect to the chassis;
- `Eng`: engine crankshaft with one rotational degree of freedom with respect to the chassis;
- `Diff`: differential cage, in line with the rear axle and with one rotational degree of freedom with respect to the chassis;
- `Bgear`: bevel gear, with one rotational degree of freedom with respect to the differential cage.

Figure B.1 shows the hierarchical structure of the model, starting from the body `"n"`, which in AutoSim is the default reference axis system fixed in space.

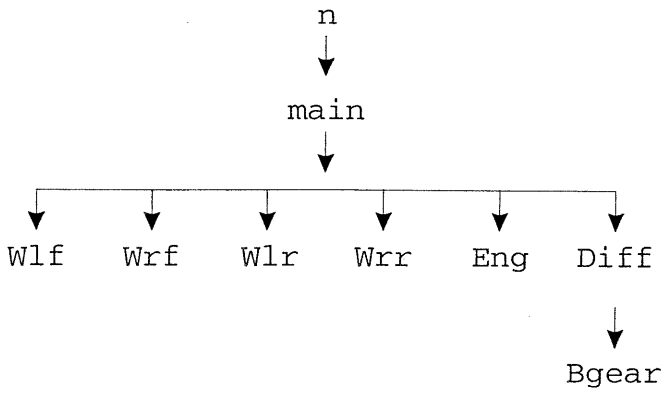


Figure B.1 Graph topology of the vehicle model.

The limited slip differential has been implemented using the methodology described by (Yeo, 1998). The degrees of freedom of the engine crankshaft, of the differential cage and of the bevel gear are removed by adding speed constraints to model the kinematic relationships between the various transmission components. Particularly:

- The rotational velocity of the differential cage is equal to half the sum of the rotational velocity of the half shafts, i.e. the rear wheels.
- The rotational velocity of the engine is equal to that of the differential cage multiplied by the total gear ratio.
- The rotational velocity of the bevel gear is equal to the difference between the rotational velocity of the half shafts.

The engine output torque is computed from the experimental engine map and is applied to the body representing the crankshaft. Next, the torque transfer due to the effect of the differential is computed as function of the differential input torque and the sign of the speed difference between the rear wheels, see Appendix A, Eq. A.7 and A.8. The amount of torque transfer is then applied to the bevel gear and is consequently transferred to the rear wheels.

Many of the vehicle subsystems have been developed separately and are coded in external C/C++ subroutines. These subroutines are included in the simulation program using the appropriate AutoSim commands. Among the quantities computed using external subroutines, also the time to distance scaling factor is evaluated. This is used to compute the actual manoeuvre time, by adding one extra state variable, using the procedure described in §6.1.2. Furthermore, all the state derivatives must be multiplied by scaling factor in order to apply the transformation. However, this was not possible to accomplish from AutoSim, and the simulation program had to be manually edited.

VEHICLE7.LSP PROGRAM LIST

```
(reset)                          ; Initialise the system
(si)                             ; Use SI units for all quantities
(add-gravity)                    ; Introduce uniforma gravitational field
(set-defaults format "'TEXT")    ; Set output in text format
(setsym *no-zees* t)
(setsym *no-pcs* t)

;; Define Multibody Name

(setsym *multibody-system-name* "7 DOF Vehicle Simulation Model")

;; Introduce vehicle body with three DOF:
;; Longitudinal displacement
;; Lateral displacement
;; Yaw rotation

(add-body main :parent n :name "Car Body" :mass M :inertia-matrix (0 0
Izz) :cm-coordinates (0 0 -hg) :translate (x y) :body-rotation-axes z
:parent-rotation-axis z)
```

```
;; Add the four wheels, each with one rotational DOF with respect to
;; the parent body and moment of inertia about the spin axes

(add-body Wlf :parent main :name "Left front wheel" :mass 0 :inertia-
matrix (0 Jf 0) :joint-coordinates (lf "-0.5*tf" -Rf) :body-rotation-
axes y :parent-rotation-axis y :reference-axis z)

(add-body Wrf :parent main :name "Right front wheel" :mass 0 :inertia-
matrix (0 Jf 0) :joint-coordinates (lf "0.5*tf" -Rf) :body-rotation-
axes y :parent-rotation-axis y :reference-axis z)

(add-body Wlr :parent main :name "Left rear wheel" :mass 0 :inertia-
matrix (0 Jr 0) :joint-coordinates (-lr "-0.5*tr" -Rr) :body-rotation-
axes y :parent-rotation-axis y :reference-axis z)

(add-body Wrr :parent main :name "Right rear wheel" :mass 0 :inertia-
matrix (0 Jr 0) :joint-coordinates (-lr "0.5*tr" -Rr) :body-rotation-
axes y :parent-rotation-axis y :reference-axis z)

;; Add engine crankshaft with one rotational degree of freedom with
;; respect to the parent body, i.e. the main body. Give it the engine
;; rotational inertia

(add-body Eng :parent main :name "Engine crankshaft" :mass 0 :inertia-
matrix (Jm 0 0) :joint-coordinates (0 0 -Rr) :body-rotation-axes x
:parent-rotation-axis x :reference-axis y)

;; Add differential cage, in line with rear axle, with one rotational
;; degree of freedom with respect to the parent body, i.e. the main.
;; No inertia properties included.

(add-body Diff :parent main :name "Differential cage" :mass 0
:inertia-matrix (0 0 0) :joint-coordinates (-lr 0 -Rr) :body-rotation-
axes y :parent-rotation-axis y :reference-axis z)

;; Add bevel gear. Its velocity is a measure of the differential
;; action, since it will rotate only as a result of a difference in
;; speed between the two rear wheels.

(add-body Bgear :parent Diff :name "Bevel gear" :mass 0 :inertia-
matrix (0 0 0) :joint-coordinates (0 0 0) :body-rotation-axes x
:parent-rotation-axis x :reference-axis y)

;; Add dynamic variables to be assigned values by external
;; subroutines. These variables must be defined BEFORE they are used
;; by any other command.

(add-variables dyvars real steer th_br tau E_trq "B_trq(4)" tt "Fa(2)"
"slip(8)" "Fz(4)" "Ftyre(8)" "trk(4)" CG_d Scf)

;; The variable tau is used in a speed constraint. Therefore AutoSim
;; needs to know its derivative. In this case, being tau a step
;; function of the velocity, we ignore the discontinuities and we set
;; the derivative to 0.0

(set-derivs tau 0.0)
```

```
;; Add set of speed constraints to remove the degrees of freedom of
;; the engine, the differential and the bevel gear. Implementation of
;; the differential rules.

;; Differential: the speed of the differential case is equal to half
;; the sum of the rear wheels rotational velocities.

(add-speed-constraint "ru(Diff) - half*(ru(Wlr)+ru(Wrr))" :u
"ru(Diff)")

;; Engine: the engine rotational velocity is that of the differential
;; cage multiplied by the total gear ratio.

(add-speed-constraint "ru(Eng) - tau * ru(Diff)" :u "ru(Eng)")

;; Bevel gear: the velocity of the bevel gear is equal to the
;; difference between the rear wheels rotational velocity.

(add-speed-constraint "ru(Bgear) - ru(Wrr) + ru(Wlr)" :u "ru(Bgear)")

;; Add subroutines from vehicle model library

;; Initialise user defined parameters and data structures. These
;; include the model parameters needed by the external subroutines and
;; the steer and throttle control tables

(add-subroutine setdef initiate() :comment "Initialise parameters for
vehicle simulation subroutine library")

;; Evaluate control parameters

(add-subroutine difeqn eval_control_variables (t steer th_br)
   :comment "Evaluate steer angle and throttle/brake position")

;; Evaluate gear ratio as a function vehicle forward speed

(add-subroutine difeqn gearbox ("tu(main,1)" tau)
   :comment "Evaluate overall gear ratio")

;; Evaluate engine output torque

(add-subroutine difeqn engine (th_br "-ru(Eng,1)*dr/6" E_trq)
   :comment "Evaluate engine output torque")

;; Evaluate differential torque transfer

(add-subroutine difeqn differential ("ru(Wlr,1)" "ru(Wrr,1)"
"E_trq*tau" tt) :comment "Evaluate limited slip differential torque
transfer")

;; Evaluate braking torque

(add-subroutine difeqn brake (th_br "&B_trq(1)")
   :comment "Evaluate braking torque")

;; Evaluate the aerodynamic forces

(add-subroutine difeqn aerodynamic_forces ("tu(main,1)" "&Fa(1)")
   :comment "Evaluate aerodynamic forces")
```

```
;; Evaluate the wheel slip quantities

(add-subroutine difeqn wheel_slip (steer "ru(main,1)" "tu(main,1)"
"tu(main,2)" "ru(Wlf,1)" "ru(Wrf,1)" "ru(Wlr,1)" "ru(Wrr,1)"
"&slip(1)") :comment "Evaluate slip quantities")

;; Evaluate wheel vertical loads

(add-subroutine difeqn wheel_load ("ru(main,1)" "tu(main,1)"
 "&B_trq(1)"  "E_trq*tau" "Fa(2)" "&Fz(1)") :comment "Evaluate wheel
vertical loads")

;; Evaluate tyre forces

(add-subroutine difeqn tyre_forces ("&slip(1)" "&Fz(1)" "&Ftyre(1)")
   :comment "Evaluate array of tyre forces")

;; Add subroutine to evaluate time to travelled distance scaling
;; factor

(add-subroutine difeqn scaling_and_position(t "rq(main,1)"
"tq(main,1)"  "tq(main,2)"  "tu(main,1)"  "tu(main,2)"  "&trk(1)"  Scf
CG_d) :comment "Evaluate state derivatives scaling factor and vehicle
position within road boundaries")

;; Add forces and moment to the system

(add-line-force F1lat :name "Left front wheel lateral force"
:direction [mainy]  :magnitude
"Ftyre(1)*cos(steer)+Ftyre(2)*sin(steer)" :point1 Wlf0)

(add-line-force F1lon :name "Left front wheel longitudinal force"
:direction [mainx] :magnitude
"-Ftyre(1)*sin(steer)+Ftyre(2)*cos(steer)" :point1 Wlf0)

(add-line-force  F2lat  :name  "Right  front  wheel  lateral  force"
:direction [mainy] :magnitude
"Ftyre(3)*cos(steer)+Ftyre(4)*sin(steer)" :point1 Wrf0)

(add-line-force  F2lon  :name  "Right  front  wheel  longitudinal  force"
:direction [mainx] :magnitude
"-Ftyre(3)*sin(steer)+Ftyre(4)*cos(steer)" :point1 Wrf0)

(add-line-force F3lat :name "Left rear wheel lateral force" :direction
[mainy] :magnitude "Ftyre(5)" :point1 Wlr0)

(add-line-force  F3lon  :name  "Left  rear  wheel  longitudinal  force"
:direction [mainx] :magnitude "Ftyre(6)" :point1 Wlr0)

(add-line-force  F4lat  :name  "Right  rear  wheel  lateral  force"
:direction [mainy] :magnitude "Ftyre(7)" :point1 Wrr0)

(add-line-force  F4lon  :name  "Right  rear  wheel  longitudinal  force"
:direction [mainx] :magnitude "Ftyre(8)" :point1 Wrr0)

(add-line-force Fax :name "Aerodynamic Drag" :direction [mainx]
   :magnitude "Fa(1)" :point1 maincm)
```

```
(add-moment E :name "Engine output torque" :direction [Engx]
   :magnitude "E_trq" :body1 Eng :body2 main)

(add-moment LSD :name "Differential torque transfer" :direction
[Bgearx] :magnitude "tt" :body1 Bgear :body2 Diff)

(add-moment T1 :name "Left front wheel torque balance" :direction
[Wlfy] :magnitude "B_trq(1) + Ftyre(2)*Rf" :body1 Wlf :body2 main)

(add-moment T2 :name "Right front wheel torque balance" :direction
[Wrfy] :magnitude "B_trq(2) + Ftyre(4)*Rf" :body1 Wrf :body2 main)

(add-moment T3 :name "Left rear wheel torque balance" :direction
[Wlry] :magnitude "B_trq(3) + Ftyre(6)*Rr" :body1 Wlr :body2 main)

(add-moment T4 :name "Right rear wheel torque balance" :direction
[Wrry] :magnitude "B_trq(4) + Ftyre(8)*Rr" :body1 Wrr :body2 main)

;; Add differential equations to evaluate manoeuvre time

(add-state-variable j jp t)
(add-equation difeqn jp "Scf" :comment "Equation for the evaluation of
the manouvre time")

;; Define output required

(add-standard-output)

(add-out steer "Steer" :long-name "Steer angle control")
(add-out th_br "Th_br" :long-name "Throttle/brake control")
(add-out tau "Tau" :long-name "Gear Ratio")
(add-out "slip(1)" "alpha1" :long-name "Left front wheel slip angle")
(add-out "slip(3)" "alpha2" :long-name "Right front wheel slip angle")
(add-out "slip(5)" "alpha3" :long-name "Left rear wheel slip angle")
(add-out "slip(7)" "alpha4" :long-name "Right rear wheel slip angle")
(add-out "slip(2)" "k1" :long-name "Left front wheel longitudinal
slip")
(add-out "slip(4)" "k2" :long-name "Right front wheel longitudinal
slip")
(add-out "slip(6)" "k3" :long-name "Left rear wheel longitudinal
slip")
(add-out "slip(8)" "k4" :long-name "Right rear longitudinal slip")
(add-out "Fz(1)" "Fz1" :long-name "Front left wheel vertical load")
(add-out "Fz(2)" "Fz2" :long-name "Front right wheel vertical load")
(add-out "Fz(3)" "Fz3" :long-name "Rear left wheel vertical load")
(add-out "Fz(4)" "Fz4" :long-name "Rear right wheel vertical load")
(add-out "Ftyre(1)" "Fy1" :long-name "Front left wheel lateral force")
(add-out "Ftyre(3)" "Fy2" :long-name "Front right wheel lateral
force")
(add-out "Ftyre(5)" "Fy3" :long-name "Rear left wheel lateral force")
(add-out "Ftyre(7)" "Fy4" :long-name "Rear right wheel lateral force")
(add-out "Ftyre(2)" "Fx1" :long-name "Front left wheel longitudinal
force")
(add-out "Ftyre(4)" "Fx2" :long-name "Front right wheel longitudinal
force")
(add-out "Ftyre(6)" "Fx3" :long-name "Rear left wheel longitudinal
force")
(add-out "Ftyre(8)" "Fx4" :long-name "Rear right wheel longitudinal
force")
```

```
(add-out j "Time" :long-name "Manoeuvre time")
(add-out CG_d "Lat_pos" :long-name "Lateral CG position")

;; Derive equations of motion

(finish)

;; Units, constants and default parameters. Remember that "hg" does
;; not enter in any calculation, hence is ignored by AutoSim.

(set-names M "Vehicle total mass"
   Izz "Vehicle yaw moment of inertia"
   Jf "Front axle polar moment of inertia"
   Jr "Rear axle polar moment of inertia"
   Jm "Engine moment of inertia"
   lf "Distance of the CG from the front axle"
   lr "Distance of the CG from the rear axle"
   half "Constant"
   Rf "Front wheel radius"
   Rr "Rear wheel radius"
   tf "Front track"
   tr "Rear track")

(set-defaults M 650 Izz 695 Jf 0.74 Jr 0.95 Jm 0.017 lf 1.846 lr
1.422)

(set-defaults tf 1.421 tr 1.408 Rf 0.327 Rr 0.326 half 0.5)

;; Write simulation code

(write-to-file write-eqs "c:\\FastLap\\autosim_mdl\\vehicle7.rtf")
(write-to-file write-c "c:\\FastLap\\autosim_mdl\\vehicle7.cpp")
(write-to-file write-h "c:\\FastLap\\autosim_mdl\\vehicle7.h")
```

Appendix C.

Analytical Evaluation of the g-g Diagram

The g-g diagram represents the maximum longitudinal, lateral and combined accelerations that a vehicle may achieve in steady state conditions, i.e. at constant yaw rate and longitudinal velocity. In order to plot the analytical g-g diagram for our vehicle model, we shall use the vehicle model equations A.1, outlined in Appendix A, to evaluate the rate of variation of some states as a function of the actual state values. The relevant equations are those for the evaluation of $\dot{x}_2$, $\dot{x}_3$, $\dot{x}_4$, $\dot{x}_8$, $\dot{x}_{10}$, $\dot{x}_{12}$, $\dot{x}_{14}$. The problem is set up as a constrained minimisation of a multivariable function. We shall assume the longitudinal velocity is fixed and equal to V.

Firstly, the maximum positive and negative longitudinal accelerations are identified. For this problem, the steer angle, the yaw rate and the lateral and longitudinal velocities are fixed:

$$\delta = 0$$
$$x_2 = x_4 = 0$$
$$x_3 = V$$

The independent variables for the optimisation are the vehicle longitudinal control and the wheel rotational velocities:

$$\mathbf{z} = \begin{bmatrix} u_{tb} & x_8 & x_{10} & x_{12} & x_{14} \end{bmatrix}$$

The problem is to find a set of values for $\mathbf{z}$ in order to maximise (or minimise) $\dot{x}_3$ while imposing the equilibrium of each wheel. In order to do so, the wheel rotational accelerations are constrained to have values which depend on the vehicle longitudinal acceleration. From Eq. A.33, taking into account that $x_2 = 0$, we may write for each wheel:

$$x_w = \frac{x_3}{R} \cdot \left(\frac{k}{100} + 1 \right) \qquad \text{Eq. C.1}$$

Here, x_w is the rotation velocity for one wheel, R its radius and k the longitudinal slip. Differentiating equation C.1 we obtain:

$$\dot{x}_w = \frac{\dot{x}_3}{R} \cdot \left(\frac{k}{100} + 1 \right) + \frac{x_3}{100 \cdot R} \cdot \left(\frac{\dot{k}}{100} + 1 \right) \qquad \text{Eq. C.2}$$

In quasi steady-state equilibrium, $\dot{k}$ is expected to be small, therefore Eq. C.2 may be approximated as follows:

$$\dot{x}_w \cong \frac{\dot{x}_3}{R} \cdot \left(\frac{k}{100} + 1 \right) \qquad \text{Eq. C.3}$$

Hence, at any solution point the values of $\dot{x}_8$, $\dot{x}_{10}$, $\dot{x}_{12}$, $\dot{x}_{14}$ are constrained to be equal to the right hand member of Eq. C.3.

Once the maximum positive and negative longitudinal accelerations have been identified, a range of values is specified within them, e.g. every 0.5 *g*. Then, for each of these values the maximum lateral acceleration has to be identified. Let *V* be the vehicle velocity and *a* the imposed longitudinal acceleration. The value of the third state variable will be fixed:

$$x_3 = V$$

The independent variables for the optimisation problem now include the longitudinal and lateral vehicle controls, the yaw rate, the lateral velocity and the wheel rotational velocities:

$$\mathbf{z} = \left[u_{st} \quad u_{tb} \quad x_2 \quad x_4 \quad x_8 \quad x_{10} \quad x_{12} \quad x_{14} \right]$$

The target of the optimisation is to find a set of values for **z** in order to maximise the total lateral force *F*:

$$F = F_{y1} + F_{y2} + F_{y3} + F_{y4} \qquad \text{Eq. C.4}$$

The set of constraints now includes the wheel equilibrium as for the previous case plus the conditions that the rate of variation of the vehicle lateral velocity and the vehicle yaw rate must be zero at the solution:

$$\dot{x}_2 = \dot{x}_4 = 0 \qquad \text{Eq. C.5}$$

Finally, the vehicle longitudinal acceleration is also constrained to have the specified value:

$$\dot{x}_3 = a \qquad \text{Eq. C.6}$$