Parameter estimation for lateral dynamic tire models using empirical inertial data

A comparative case study of gradient-based algorithms vs genetic algorithms

D.H. den Hartog

Master of Science Thesis

TU Delft Delft University of Technology

Vehicle Engineering

Parameter estimation for lateral dynamic tire models using empirical inertial data

A comparative case study of gradient-based algorithms vs genetic algorithms

Master of Science Thesis

For the degree of Master of Science in Vehicle Engineering at Delft University of Technology

D.H. den Hartog

April 17, 2018

Faculty of Mechanical, Maritime and Materials Engineering (3mE)
Delft University of Technology

Delft University of Technology
Department of
Vehicle Engineering (ve)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Parameter estimation for lateral dynamic tire models using empirical inertial data

by

D.H. den Hartog

in partial fulfillment of the requirements for the degree of

Master of Science Vehicle Engineering

Dated: April 17, 2018

Supervisor(s):

Dr.ir. R Happee

Dr.ir. B. Shyrokau

Reader(s):

Dr.ir. A.L. Schwab

Abstract

Self-driving vehicles are the future of automotive engineering. Systems that take over control from the driver are developed to be able to interact with the conditions of the road and other obstacles. To develop these systems, developers use vehicle models to simulate the behaviour of the moving vehicle. The systems developed using these models are transferred the vehicle and are expected to perform accordingly. Therefore it is important that the vehicle models reflect the behaviour of the actual vehicle.

This thesis investigates the methods used to estimate the model parameters which influence the resulting vehicle simulation. Experimental tests were carried out to acquire vehicle body inertial measurement data. This data was used to minimize lateral acceleration and yaw-rate error produced by the output of a single track vehicle model using a simplified Magic formula developed by H. Pacejka. Deterministic gradient-based algorithms, such as multi-objective Sequential Quadratic Programming (SQP), are frequently used as a way to optimize an initial approximation of the model parameter set. It is shown that this initial parameter strongly influences results of the algorithm due to local minima.

As an alternative, genetic algorithms were investigated to minimize the influence of the initial parameter set. An adaptive component, Temporal Difference Q Learning (TDQL), was also added to further reduce the input of the user and to increase the performance of the algorithm.

Four algorithms, of various complexity, were implemented in Matlab and executed. The performance of the resulting parameter sets, as well as the performance of the algorithms themselves, were analysed and compared. It is concluded in this thesis that the adaptive genetic algorithm performs slightly better than a simple gradient-based algorithm with respect to the objectives and the duration of the algorithm. However, a genetic algorithm without TDQL is recommended for its good performance and the simulation flexibility of the results for this simple vehicle and tire model.

Table of Contents

List of Figures

List of Tables

Preface

I have always gravitated towards systems that move while growing up. So it is befitting that I would do finish my studies as an automotive engineer. Doing this thesis pushed me to change my Master's track to Vehicle Engineering from Precision and Microsystems Engineering even though I was already doing the automotive specialisation. This Thesis, which started out as an exploratory study of the Toyota Prius and the making of a simulation platform, has given me a lot of new experiences when it comes to hands-on vehicle measurement and working with very interesting optimization algorithms. Algorithms that go much further than only vehicle engineering and can be applied to anything.

I would like to thank Barys Shyrokau as my supervisor for his expertise and patience while helping me through this Thesis. An acknowledgement to Tom Dalhuisen for his help during the experimental testing with the Prius. A special thank you to Riender Happee and Arend Schwab for being part of my thesis committee.

Furthermore, I would like to thank my Parents, Sister and Geert-Jan for their presence and love during my thesis and studies, through the thick and the very thin. My friends, for giving me the best time a student can have and an acknowledgement to Christopher Magan and Dennis van der Arend for the motivation and company on campus. Lastly I salute Delft, a beautiful city that has stolen my heart.

Dedicated to

Hans B. Pacejka

Chapter 1

Introduction

1-1 Background

The direction of developments in the automotive world is changing rapidly to accommodate the demand of sustainable systems, a higher efficiency of transport and technological improvements in general. Self-driving vehicles have always been a large part of society's expectation of the future. With the current technology where it is, this looks to become a reality. Currently, there are already vehicles on the roads of Europe which are able to control its own operations to a certain extent with systems such as lane keeping and adaptive cruise control. Manufacturers of cargo transport vehicles are also developing self-driving vehicles to relieve drivers and to increase the capacity of the roads.

There will be a point when the majority of the vehicles on the road are automated to drive on their own. It is therefore important that the infrastructure, and vehicles themselves, are able to communicate and work with each other. The position and motion of every vehicle will be the most important piece of information shared. This will most likely come from real-time Global Positioning System (GPS) signals and vehicle state measurement systems already installed in most modern vehicles. The measurement data from these systems may be used to develop accurate models to predict vehicle behaviour for current and future driving situations [4].

Driving control systems are developed based on vehicle and tire models. The accuracy of these models depends on its design and the model parameters. Every vehicle and tire manufactured has its own handling characteristics and therefore a different set of model parameters. Obtaining these parameters may be difficult as information is kept private or are highly priced by the manufactures and testing institutions. This forces developers, such as those at universities, to reverse engineer relevant vehicle components and their characteristics. These are used to create and tune the vehicle models used in system development and research.

Vehicle and tire parameters are frequently roughly estimated or taken from models of similar vehicles and further tune to produce the desired model results. Parameter estimation using optimization algorithms are used but due to the rough estimate, the output can contain error

or will take a lot of effort to tune. With sufficient measurement data from the vehicle, this process can retraced and the optimization algorithms investigated. This forms the focus of this thesis.

1-2 Thesis and Research question

The Delft University of Technology (TU Delft) has a Toyota Prius available as a test vehicle for, but not limited to, the development of vehicle control systems. The vehicle is equipped with measurement systems to accurately record the state of the vehicle as well as its position by means of a Global Navigational Satellite System (GNSS) receiver which also records the vehicle's inertia and motion.

Experimental manoeuvres were performed and the recorded data was used is to validate the output of vehicle models and the incorporated tire model. Usually parameters are estimated based on available knowledge and are later hand-tuned to achieve accurate results. To reduce the duration of this iterative task, optimization algorithms are implemented.

Deterministic algorithms produce model parameters by iterations based on characteristics of the data. This may include the model error and the gradient of that error with respect to change in parameter values. The deterministic algorithms used in this study will primarily be gradient based and requires an initial parameter set to be estimated as a starting point for the optimization. A stochastic algorithm does not require an initial estimate of the parameters as it is based on logical programming using a random search method.

This study aims to compare deterministic algorithms with stochastic algorithms with the goal to determine if the latter performs better in terms of accuracy, reliability, flexibility and computational efficiency. A better performing algorithm provides the option to increase the model's complexity as well as the ability to process larger datasets. Larger datasets provide more information about the vehicle and therefore will be able to develop a more accurate model.

Furthermore, the aim of this study is to reduce the number of variables that need to be pre-defined by the user. This reduces the uncertainty of the model's outputs due to the sensitive algorithm parameters that may influence its output. These goals yield the main research question for this thesis:

> Does an adaptive genetic algorithm perform better at estimating tire model parameters than gradient-based algorithms which use pre-defined initial tire model parameters?

Thesis structure

This thesis report is structured according to the to the work-flow of the research as follows:

- Chapter 2 reviews vehicle dynamics theory and forms a basis for the building of vehicle and tire models. This chapter also gives an explanation of the parameters and outputs that will be used by the optimization algorithms.
- Chapter 3 will cover the retrieval of measurement data from the test vehicle. This includes explanations of the measurement instruments, experimental manoeuvres and recorded data characteristics. The focus will be placed on how the data will be used in later model optimization.
- Chapter 4 will cover in detail the deterministic and stochastic algorithms used in this study. Explanations and illustrations will be given on the theory, processes and the implementation of these algorithms in Matlab. The chapter also includes implementation of the algorithms to ensure the results of these algorithms are comparable.
- Chapter 5 will cover the results of the model optimization using the different implemented algorithms. The performance of these algorithms will be evaluated as well as the simulation output based on the resulting parameter solutions. The characteristics of the search space will also be investigated as well as other aspects of optimization relevant to answering the research question.
- Chapter 6 contains the conclusion of this study as well as recommendations for future work.

Chapter 2

Vehicle dynamics

The theory behind vehicle dynamics is based on simple free-body mechanical systems. This can be made more complex depending on the vehicle and the purpose of the model. Simplifications of specific vehicle models are chosen to accommodate the limitations of the experimental data as well as the availability of technical information on the vehicle. Limitations associated with the duration of this study are also a factor in the choices made during the building of models.

The first section of this chapter will describe the theoretical contributions made by the literature on this subject. The second and third section will cover the tire and vehicle models while the fourth section is written as a precursor to the parameter estimation. The limits and characteristics of the model that need to be optimized be explained. This includes the model inputs and outputs. The last section concludes the chapter with an explanation of the model's implementation in Matlab as well as a discussion regarding the finished model and the choices made in its construction.

2-1 Literature contributions to this study

Many textbooks such as *Vehicle Dynamics and Control* by Rajamani et al. [1] and *Vehicle Handling Dynamics* by Abe et al. [3] explain vehicle handling dynamics with a goal to model vehicle behaviour needed for automated control systems. For this study, these books provide the basis for the single track model, transient dynamics and load transfer used.

While vehicle body dynamics can be considered straightforward multi-body dynamics, tire dynamics are much more difficult to model. Therefore many models were created by engineers to achieve this with various levels of accuracy and computational efficiency. Each model having its own advantages and disadvantages when it comes to the type of manoeuvres that it is able to simulate.

Hans Pacejka, a researcher at the TU Delft developed the magic formula which is widely used in vehicle modelling to this day [2]. There are many publications by Pacejka and other

researchers that have expanded and supplemented this initial semi-empirical model. Most of the theory presented in this paper will be based on a simplified model version of these works.

2-2 Tire dynamics

The tires largely determine the handling characteristics of the vehicle as they produce the lateral and longitudinal forces experienced by the chassis. These forces originate from the tire contact patch, which is the area of contact between the tires and the road surface. Forces are then transmitted through the tire structure to the solid wheel rim. Eventually further through the suspension towards the vehicle's chassis. Tire dynamics cover the force produced by the contact patch and the deformation of the tires under the load along with the steering and acceleration of the vehicle.

A tire is constructed out of many layers of material, moulded together to make up the side walls, tread and interior components. This makes a tire very anisotropic which means that it exhibits non-linear dynamic material behaviours. These are influenced by independent factors such as tire wear, pressure and temperature [5]. How the tire is set up to interact with the road also determines much of the handling characteristics. These operating factors include tire load and its camber, toe, and caster angle.

While a complete physical model may be used to describe these tire dynamics, application of these methods is very complex. It consumes a lot of time and processing power. Therefore, other models are developed to describe these behaviours in a less complex way in order to avoid long computational times. Some have been developed to do this by modelling the tire composition with other components with simple dynamic principles. Examples are the brush model, linearised spring/damper models, which use the principles of rigid beam deformation, and springs.

Empirical models are based on the analysis of a large number of experimental datasets investigating the behaviour of specific tire characteristics. These results reveal a set of reproducible patterns which can be fitted by the outputs of a formula which can be used in a larger model. The application of these models is limited to tire characteristics investigated to create the model as there are minimal physical principles in place to accommodate situations outside this scope.

Input and outputs (shown in table 2-1) describe a complete tire model encompassing lateral, longitudinalforces and the various moments experienced by the tire. For all outputs to be accurate, this accuracy must be reflected in the measurement data of the inputs. As not all inputs of the table are accurately available, assumptions must be made to reduce the inaccuracies of the outputs.

Power-train information such as wheel speed and torque, are primarily used to calculate the tire's slip-ratio. This is done with respect to the road surface, and as the power-train forces are currently not obtainable, this makes modelling longitudinal dynamics not feasible for this study. While longitudinal slip does affect the forces produced by lateral slip, it is assumed in this study that this effect is minimal.

Other characteristic outputs of the model such as aligning torque, resistance moment and overturning couple are also affected by the similar lack of measurement capabilities. The input values for these behaviours are assumed to be zero.

Table 2-1: Inputs and outputs of a complete tire model

Input vector		Output vector	
ρ	Radial deflection	F_z	Normal load
κ	Longitudinal slip	F_x	Longitudinal force
Ω	Angular velocity	M_y	Rolling resistance moment
α	Slip angle	F_y	Cornering force
ϕ	Turnslip	M_z	Aligning torque
γ	Camber angle	M_x	Overturning couple

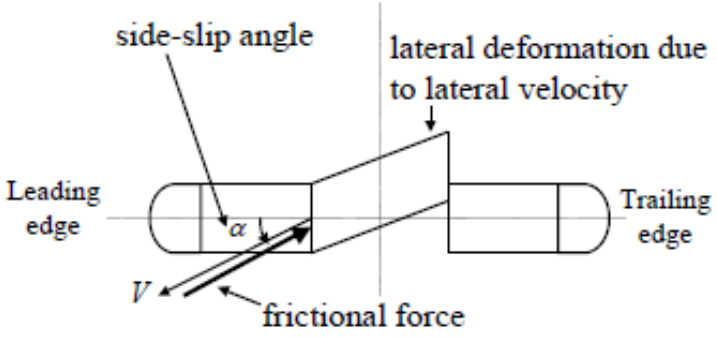


Figure 2-1: Contact patch of a tire [1]

Lateral vehicle dynamics are largely determined by the forces produced within the area of the tire in contact with the road surface, called the contact patch. Figure 2-1 illustrates the deflection experienced by the tire during a cornering manoeuvre. It also shows the leading edge where the rubber of the tire deforms and does not experience slip. The trailing edge is where the rubber of the tire slowly detaches from the road surface and experiences a relative slip. As the tire turns and moves forwards, a single point on the tire tread will go through a complex dynamic deflection and experience friction which are difficult things accurately model [1] [3].

The slip angle is the angle between the heading of the tire and the tire's deflected leading edge as seen in figure 2-1. This angle is essentially the direction the tire velocity relative to the heading of the wheel.

$$\alpha_f = \delta_{steer} - \tan^{-1}\left(\frac{v_y + l_f\ \dot{\Psi}}{v_x}\right) \tag{2-1a}$$

$$\alpha_r = -\tan^{-1}\left(\frac{v_y - l_r\ \dot{\Psi}}{v_x}\right) \tag{2-1b}$$

Equations 2-1a and 2-1b calculate the front and rear tire slip angle α_f and α_r measured in radians. These are calculated using the global motions measured at the centre of gravity of the car. The total lateral velocity of the front and rear tires is a combination of the global lateral velocity of the vehicle v_y and the lateral velocity produced by the yaw rate $\dot{\Psi}$ multiplied by the position of the wheels l_f and l_r with respect to the position of the centre of gravity.

The longitudinal velocity in the direction of the tires is assumed to be the general velocity of the vehicle due to the small angles involved.

The velocity angle of the tires with respect to the orientation of the tires can be calculated to produce the slip angles of each tire. As the front wheels are able to turn to steer the vehicle, this steering angle δ_{steer} is compensated to ensure that the orientation definition of the slip angle is kept constant.

$$F_{yf} = 2K_f\alpha_f \tag{2-2a}$$

$$F_{yr} = 2K_r\alpha_f \tag{2-2b}$$

Initially, an assumption can be made that the tire lateral forces interact linearly with the slip angle and can be calculated with (2-2). However, the condition of this assumption is that this is only accurate for simulations where the side-slip angles are small. When simulating manoeuvres that exhibit large side-slip angles, a non-linear behaviour is observed which produces large errors when modelled with a linear tire model. This is shown in figure 2-2 where two manoeuvres are plotted showing the linear models inability.

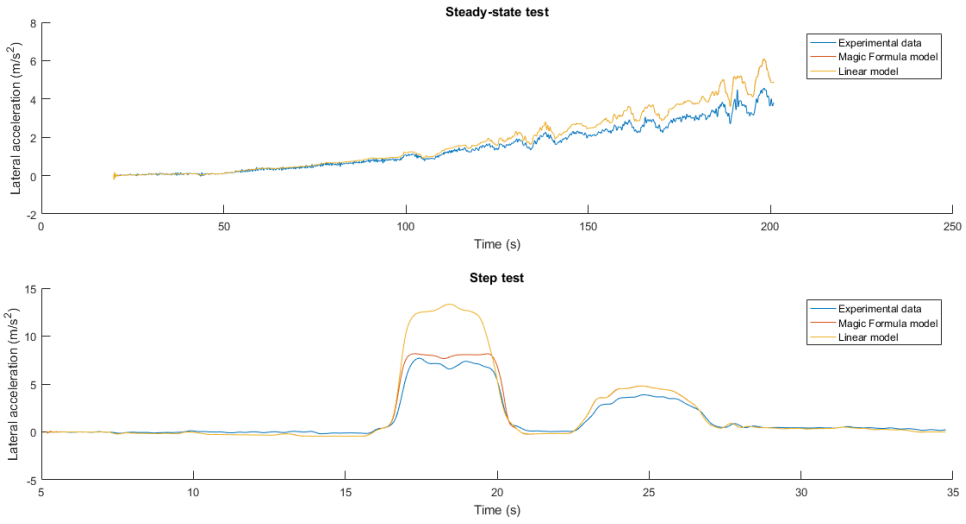


Figure 2-2: Linear vs non-linear magic formula tire model showing the inability of a linear tire model to cope with extreme lateral acceleration

The upper figure shows the vehicle at a constant steering angle and increasing speed. The slip increases with velocity until the linear model starts to simulate larger lateral tire forces than is measured in reality. This effect is also observed when applying the model to dynamic steering manoeuvres as seen in the lower plot of figure 2-2. The first steering plateau seen in the plot is executed at a higher longitudinal velocity and therefore slip angles while the second is performed at a lower velocity thus remaining within the linear tire limits. It is therefore essential to use non-linear models when all driving situations are expected to occur.

2-2-1 Magic formula

The Magic formula is a semi-empirical tire model developed by a collaboration of TU-Delft and Volvo [2]. It is developed using experimental data compiled by performing experiments in laboratory conditions. The resulting formula is applied widely by developers in their research into vehicle systems and control such as the works of [6] [7] [8].

$$y(x) = F_z D\Big(C \arctan\left(Bx - E(Bx - \arctan(Bx))\right)\Big) \tag{2-3}$$

Equation 2-3 represents the basic form of the Magic formula. The formula can be used to calculate lateral and longitudinal tire forces as well as aligning torque, this is done by substituting F_y, F_x or M_z in the place of y. Input x can be substituted by the slip angle α and the slip ratio κ for the respective output. The scalar coefficients B, C, D, and E are user defined and are varied to fit the formula to a model tire and working conditions.

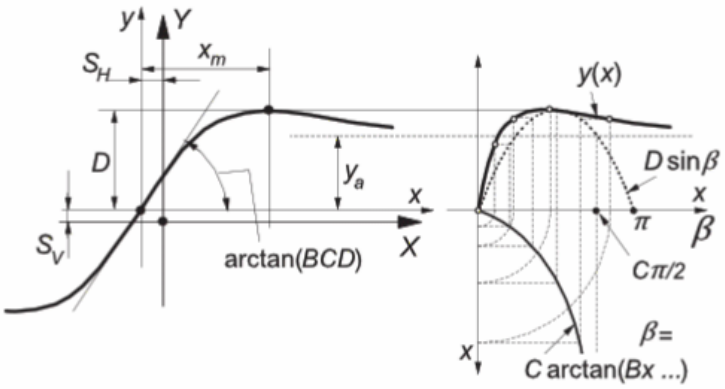


Figure 2-3: Illustration of the magic formula graph [2]. The x-axis signifies the side-slip angle and the y-axis signifies the force produced by the tire.

The resultant curve of the Magic formula, as shown in 2-3, intersects the origin and increases linearly with slip until it starts to saturate and reaches a peak. This is the maximum friction obtainable by the tire. The gradient of the initial part of the curve, which is also called the linear cornering stiffness, can be calculated by multiplying BCD. The D coefficient describes the friction coefficient between the road and tire and is multiplied with the normal tire load F_z which produces the maximum achievable lateral tire force according to the principles of friction. The C coefficient determines the shape of the curve, this value found to lie between 1.2 and 1.6. This leaves the coefficient B which varies the previously mentioned cornering stiffness BCD. The coefficient E is used to determine the shape of the maximum of the curve and to vary the slip value where the maximum occurs. The curve then decreases and moves towards a horizontal asymptote describing the dynamic friction force present when the majority of the contact patch starts to slip across the road which produces lower lateral forces than produced static friction.

Due to irregularities in the materials of the tire, it is possible that a force is produced when no tire slip is present. To compensate for these phenomena the curve can be horizontally

and vertically shifted using the parameters S_h and S_v respectively. These parameters are implemented into the Magic formula using (2-4).

$$X_s = X + S_h \tag{2-4a}$$

$$Y(X) = y(X_s) + S_v \tag{2-4b}$$

This extended tire model shown in appendix A-1 takes other characteristics into consideration such as the camber angle, tire temperature, pressure and vertical load. The basic Magic formula is still used but the parameters are further split into further detailed parameters related to these additional characteristics. This extended version is most likely used by tire modelling software such as Delft Tyre developed by the Netherlands Organisation for Applied Scientific Research (TNO). This is important as the initial parameters used in this study are derived from tire files using parameters used by the extended version of this formula.

2-2-2 Relaxation length/Transient dynamics

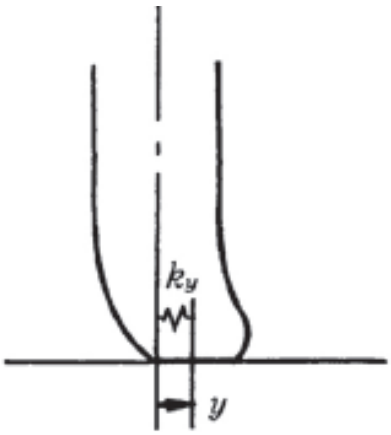


Figure 2-4: The tire deformation during a cornering manoeuvre where k_y is the lateral stiffness of the tire. [3]

The magic formula is a steady-state model and as such will be less accurate during the moments before a steady-state is achieved. To compensate for this inaccuracy during dynamic manoeuvres, additions to the tire model are made which take these transient effects into account. This modification takes places after the Magic formula has calculated the lateral tire forces.

The sides of a tire are made of rubber which can be modelled as a lateral spring and damper system. This model's the deformation of the tire caused by lateral forces originating from the tire contact patch. Translating this simple model for vehicle dynamics produces the relaxation length which is the distance a tire needs to travel before the tire has finished deforming to accommodate the new forces applied by the motion of the vehicle. This can be implemented into the vehicle model as a delay produced by (2-5).

$$\tau_y \dot{F}_y^D + F_y^D = F_y^S \tag{2-5a}$$

$$\tau_y = \frac{RL}{v_{cp}} \tag{2-5b}$$

where F_y^S is the steady-state force produced by the vehicle model, F_y^D is the dynamic lateral force which is the lateral force with consideration of relaxation length and a lag coefficient τ_y [9]. Equation 2-5b shows how τ_y is calculated using the relaxation length RL and the velocity of the tire going through the contact patch v_{cp}, at that point in time. This means that the response of the tire to sudden steering change will improve as the vehicle velocity increases.

2-3 Lateral single track model

A road going vehicle contains many complex moving parts such as the tires, wheels, suspension, chassis, drive train and steering system. For the ultimate vehicle model, all these components need to be modelled and be made to work together to globally simulate the vehicle's behaviour on the road.

The single track model is a simplified model that makes the assumption that the left and right wheels behave the same and thereby reducing the model to a vehicle with a front and rear wheel. This model is frequently referred to as the Bicycle model. This assumption reduces the accuracy of the model as it disregards the differences in tire steering angle, velocity and the lateral load of the tires during a cornering manoeuvre. However, these reductions in accuracy have no major effect on the overall accuracy. This is proven by the fact that it is widely used as a valid simplified vehicle model and it taught as such.

$$m(\overbrace{\ddot{y} + \dot{\Psi} v_x}^{a_y}) = F_{yf}(\alpha_f) + F_{yr}(\alpha_r) \tag{2-6a}$$

$$I_z \ddot{\Psi} = l_r F_{yf}(\alpha_f) - l_f F_{yr}(\alpha_r) \tag{2-6b}$$

Figure 2-5 illustrates the simplification and provides a basis for the equations of motion (2-6). The motion of the vehicle's body is calculated around the centre of gravity (P in the illustration) which represents the position of the total mass of the vehicle m. Both front and rear tires provide a lateral force to the body which results in the vehicles total translational or rotational motion about P. The two outputs of this model are the inertial acceleration and the yaw rate $\dot{\Psi}$.

The inertial lateral acceleration a_y, calculated using the equation of motion (2-6a), consists of two components being the acceleration due to movement along the y-axis $\ddot{y}$ and the centripetal acceleration due to the rotation of the vehicle about the path radius.

The yaw rate $\dot{\Psi}$ results from the difference in the total moment applied to the vehicle body by the tires. Equation (2-6b) is a rotational equation of motion that calculates the change in yaw rate $\ddot{\Psi}$ by means of the vehicles moment of inertia I_z. The $\ddot{\Psi}$ is differentiated to produce the yaw rate $\dot{\Psi}$ which is used as an output in this study alongside the lateral acceleration.

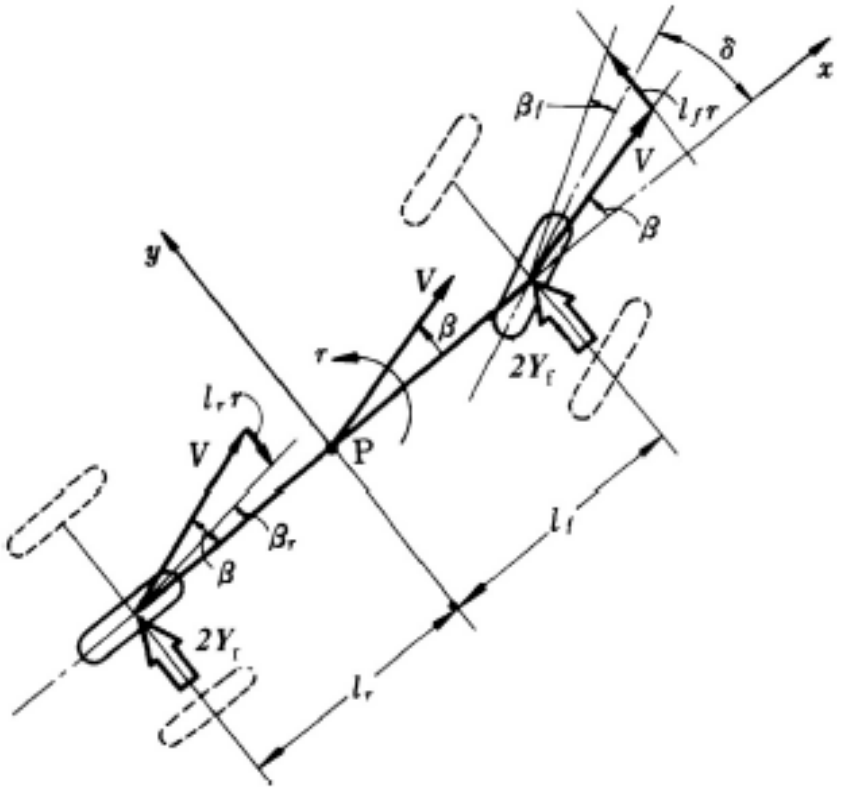


Figure 2-5: Illustration of the single track model [3]

2-4 Longitudinal load transfer

The single track model is called a planar model as it only models motion and forces in the lateral and longitudinal directions. As described in section 2-2-1, the vertical load on the tires significantly affects its performance and is, therefore, an important aspect of vehicle dynamics to take into account. This is especially true when the longitudinal velocity of the experimental data is not constant. The changes in longitudinal velocity cause the centre of gravity to experience an acceleration. As the centre of gravity does not lie on the same vertical plane as the wheels, a load transfer takes place due to the conservation of momentum.

$$\Delta F_z = \frac{a_x m h_c}{l} \tag{2-7}$$

Equation (2-7) is the tire load difference for which a positive longitudinal accelerationa_x ensures that the rear tires experience an increase in tire load as a function of the height of the centre of gravity h_c, total length of the wheel base l and the vehicle mass m. Respectively, the front wheels experience a negative change in tire load when a positive longitudinal acceleration is present.

This change in normal load F_x on a tire affects the maximum amount of a lateral force it can produce as explained in section 2-2-1. It also affects the overall lateral force that a vehicle

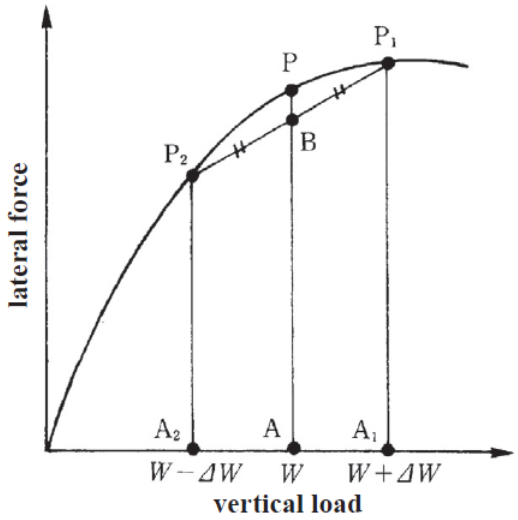


Figure 2-6: The tire force consequence of load transfer during acceleration where P is the force produced by either tire during zero load transfer and P_1 and P_2 represent the lateral force for the front and rear wheels respectively during load transfer [3]

can produce while manoeuvring a corner as one tire experiences an increase in lateral force and the other experiences an even greater loss of lateral force. The resultant force B of P_1 and P_2 during load transfer, as seen in figure 2-6, is collectively lower than its original value P. This shows that load transfer reduces the overall handling performance of a vehicle and is important to model as a model shouldn't simulate forces that do not exist in reality.

2-5 Complete model and implementation

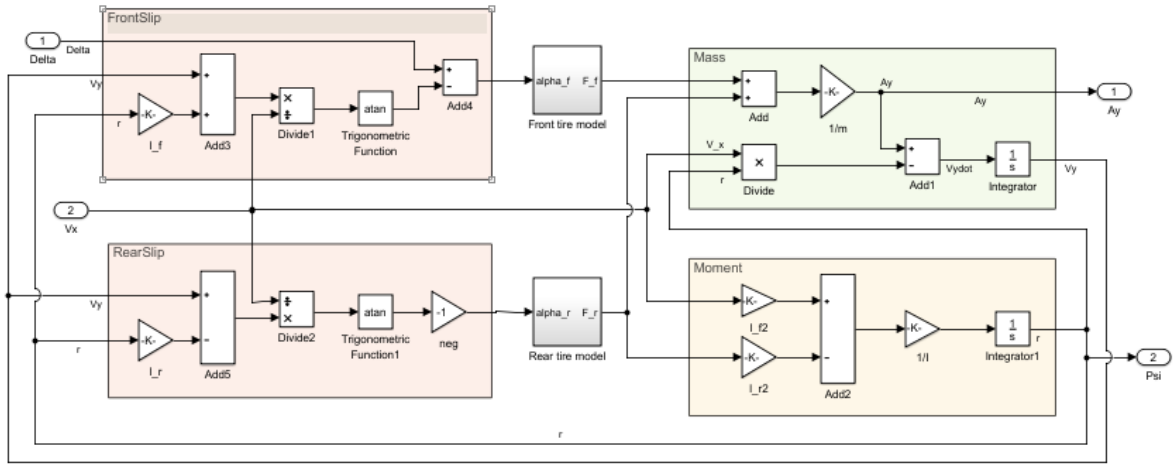


Figure 2-7: Visual model of the vehicle dynamics implemented in Simulink

The vehicle and tire models are implemented in Simulink, which is a block diagram environ-

ment that is able to simulate the behaviour of the models. This was used because Simulink efficiently builds models during the development phase despite it taking relatively longer to simulate relative to if the models were implemented in Matlab.

The steering wheel angle, longitudinal velocity and acceleration are the model inputs and the lateral acceleration and the yaw rate are the outputs that will later be used to compare to the experimental data. The implementation of the vehicle model is illustrated in figure 2-7 where the slip calculation are shown in red, the translational and rotational dynamics are shown in green and yellow respectively.

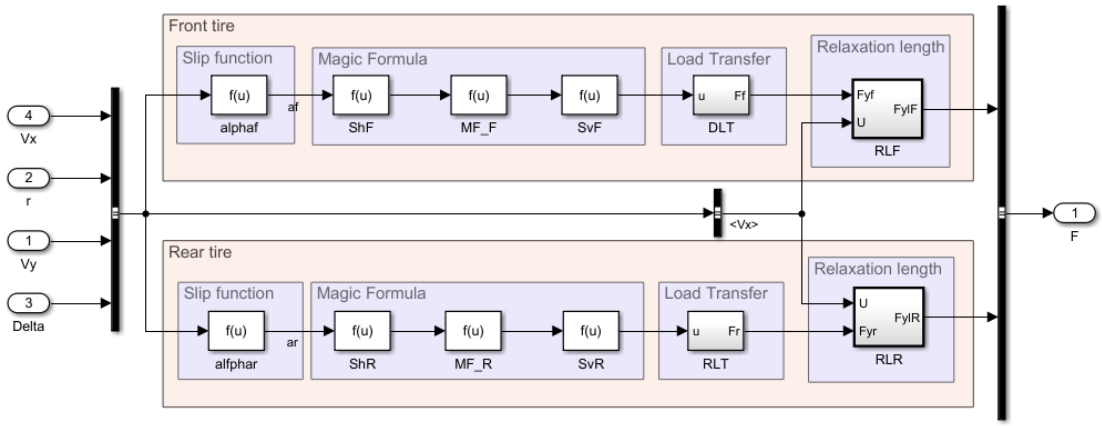


Figure 2-8: Visual model of the tire dynamics implemented in Simulink

The tire model is shown in figure 2-7 as subsystem and is expanded in figure 2-8. The inputs of this subsystem, found to the left of the figure, include the two outputs, lateral acceleration and yaw rate from the previous time stamp, as well as the steering angle and longitudinal velocity. The signal is split in such a way that the front and rear wheels are modelled independently to give the overall vehicle model the freedom to adapt to differences in suspension, tire wear and wheel set up not taken into account by the vehicle model. This drastically increases the number of parameters that need to be optimized however the accuracy gained is worth the extra computational power required.

The tire slip angle is calculated first followed by the functions related to the Magic formula tire model. The *MF* block holds most of the tire parameters while the curve shift functions, placed on both sides of this function, contain the last two parameters. The tire model outputs the friction coefficient which is multiplied by the net tire load calculated by the load transfer function. Finally, the resulting lateral force values are modified by the relaxation length function to produce the lateral tire forces used in the rest of the vehicle model. The relaxation length function contains the last two parameters. The relaxation length is unknown and can not be easily measured, this is why it is being included in the parameter optimization. It also gives the model a greater ability to adjust to non-linear changes found in the experimental data and does not have to compromise on other parameters.

$$\mathbf{X} = \begin{bmatrix} D_f \\ C_f \\ B_f \\ E_f \\ Sh_f \\ Sv_f \\ D_r \\ C_r \\ B_r \\ E_r \\ Sh_r \\ Sv_r \\ RLf \\ RLr \end{bmatrix} \tag{2-8}$$

Vector $\mathbf{X}$, show in (2-8), contains all the parameters mentioned and are varied within their respective function blocks during the parameter optimization. These are by far not the only uncertain parameters present within a vehicle model. Parameters such as the vehicle mass, moment of inertia and centre of gravity can be found in literature but may vary slightly with the specific vehicle used in this study. With the exception of the moment of inertia, most vehicle parameters can be directly measured and input into the model. Magic formula tire files do exist, however, the tire parameters in these files describe a tire with a specific condition. Tire characteristics change during its operating life and therefore may differ in reality to that of the tire files. The parameters also tend to vary per vehicle and driving conditions.

2-6 Model limitations

As previously mentioned, the vehicle and tire models need to be simplified. It is done to later be used in parallel with the data and equipment available to properly estimate the parameters needed for an accurate model.

There are a few assumptions that have to be made when using this particular model. It is a single track model which means it is assumed that the effects due to roll and that of the suspension are negligible and therefore not modelled. The Magic formula in itself is a steady-state tire model. However, with the modification of the transient dynamics in the form of a relaxation length, the model can be used to model simple dynamic manoeuvres.

That being said, this model may show inaccuracies when the conditions of the tires start to change such as the heat generated during cornering and the increase in tire pressure that follows. For this reason, this model will not be able to simultaneously produce accurate results for two very different models consisting of different changes in vehicle states that are not covered by the tire and vehicle model.

Chapter 3

Experimental data retrieval

Experiments were conducted using the Toyota Prius of the Delft University of Technology (TU Delft) which is equipped to measure its position, movement and state. The executed experimental tests executed consist of a steady-state and step response manoeuvre which provide a broad database for optimization.

The first section reviews the Toyota Prius to determine the general characteristics which may influence the deciphering of the measured data. The next sections cover the measuring equipment, experimental manoeuvre and procedures used during the experiments. Finally, the measured data is analysed and processed in preparation for optimization algorithms to be implemented.

3-1 Test vehicle

The available test vehicle at the university is a 2010 third generation Toyota Prius which is a mid-sized hybrid sedan. It is considered the personal vehicle that started the hybrid revolution in the automotive industry and set a benchmark for many vehicles of its type. Although being a hybrid is not relevant for this study, a consequence is that the vehicle carries heavy batteries that have an influence on its weight characteristics compared to other cars in the same size category.

The steering of the Toyota Prius is a rack and pinion system with Electronically assisted Power Steering (EPS) attached to the steering column and has its own electrical control unit which adjusts the steering assistance relative to the velocity of the vehicle. The speed of the vehicle is measured at the wheels and is also used by other systems such as vehicle stability control and ABS. The EPS provides the ability to turn the steering wheel electronically [10].

The technical information in table 3-1 were taken from the Toyota's technical publication on the model [10]. While this resource does contain the weight information of the vehicle, due to the measurement equipment installed, the vehicle was weighed at a calibrated weighing station with a platform accuracy of $\pm$10kg. The weight measurement was taken with a full

Table 3-1: Technical information used in vehicle models and simulation

Parameter	Value
Wheel base	2.7 m
Steering ratio	15.1
Moment of inertia	2400 kgm^2

tank and with the same amount of passengers that were to be on-board during the tests. The result of this measurement is that the vehicle weighs 1590 kg. As both axles were measured, the position of the centre of gravity is calculated to be 1.09 m behind the front axle. The measurements can be found in appendix B.

3-2 Measurement instruments

The two main measuring systems are the Spacial Dual by Advanced Navigation and the Move-box by the Netherlands Organisation for Applied Scientific Research (TNO) which serve the goals of this study to measure global motion and to retrieve vehicle state information respectively. Inertial information is measured by both systems and therefore an analysis needs to be made between to choose the most reliable source of measurements.

3-2-1 Spacial Dual

The Spacial Dual utilizes the combination of outputs from a Global Navigational Satellite System (GNSS) and is similar to the American Global Positioning System (GPS). It uses signals from multiple satellites to triangulate the position and velocity of two antennae attached to the roof of the vehicle. The accuracy of this system is increased by introducing differential error filters such as Real-time Kinematic System (RTK) navigation that use phase measurements from ground base stations to reduce the GNSS error. With these systems, the accuracy of the Spacial Dual can be reduced to as low as 0.008 m from the original 1.2 m.

One of the biggest disadvantages of a system such as RTK is that it needs to be in the line of sight of a base station. This is also the case for the GNSS signal with respect to the minimum number of satellites. The high accuracy of the GNSS and RTK system may be momentarily interrupted and therefore long-term accuracy is not assured. To compensate, an Inertial Navigation System (INS) measures accelerations and calculates the integrals to determine

Table 3-2: Maximum accuracy of the Spacial Dual measurements

Input	Accuracy	Unit
Timing	20	ns
Position	1.2	m
Position (RTK)	0.008	m
Velocity	0.007	m/s
Heading	0.1	deg
Acceleration	0.001	m/s^2

Table 3-3: The outputs of the Move-Box

Input	Resolution	Unit
Epoch Time	1	s
Throttle pedal position	0.1	%
Brake pressed	T/F	NA
User steering torque	0.1	Nm
Longitudinal Velocity	0.01	m/s
Longitudinal acceleration	0.01	m/s^2
Lateral acceleration	0.01	m/s^2
Yaw-rate	0.002	rad/s
Steering wheel angle	0.01	rad
Wheel speed	0.01	m/s

velocity and position. The measurements made by the accelerometer are not accurate for a long period of time due to noise and drift experienced by the accelerometer.

These measurements are used as a supplement and are refreshed once accurate GNSS measurements return. An internal filter removes accuracy fluctuations and determines which measurement is more accurate. The measurement module which contains the INS sensors is located inside the vehicle, roughly 15 cm behind the actual centre of gravity [11].

Having two antennae on the roof of the vehicle enables the Spacial Dual to measure the position and velocity at two points giving it the ability to calculate its heading and yaw rate to an accuracy of 0.1 deg.

3-2-2 Move-box

The Move-Box is an interface system developed by TNO to collect information as well as control low-level vehicle systems. The Move-Box is connected to the Engine Control Unit (ECU) of the Toyota Prius where it requests, receives and sends Controller Area Network (CAN) bus messages at a rate of 25 Hz. These messages contain information on the state of the vehicle such as steering wheel angle, velocity and information such as the throttle position, brake pedal switch and acceleration information from the build in accelerometer.

This system is considered to be a black box as there is little information on how the measurements are being made. Throttle position, steering wheel angle and the velocity measurements are assumed to be accurate as they are measured directly at the source and the resolution is available in the systems documentation. Measurements from the accelerometer are considered less accurate as there is no information on the sensor and its position in the vehicle. An overview of the Move-Box output is provided in Table 3-3.

3-3 Experiments

The tests were carried out on the 30th of January 2017 at the hanger terrain of Valkenburg airport. The road surface is a mixture of tarmac and cement slabs as seen in figure 3-1 which shows the test vehicle moving towards the concrete slabs. The cement slabs provide a relatively flat surface but contain irregularities in the form of the slab borders which will likely provide high-frequency disturbances to the measurements.

The asphalted concrete does not have these irregularities but the surface is very uneven. The width and breadth of the tarmac were not large enough to perform certain experiments up to International Organization of Standardization (ISO) standards. The weather conditions on the day of testing were considered wet during the steady-state test but were in the process of drying during the execution of the step response test. This might show a difference in the tire parameters, especially the road friction coefficient D.

Just before the test date, the vehicle went through its regular maintenance. The test site is 27 km from the TU Delft campus and is considered to result in sufficient driving time to bring the vehicle and its tires up to normal working temperature. All tests are initiated when the vehicle is stationary to give the measurement equipment time to initialize and calibrate.

Both the Move-box and Spacial Dual must be initiated sequentially and the timestamps later synchronized. A sudden impulse of the throttle before the tests begins acts as a benchmark for synchronisation which is done by the method of cross-correlation. The longitudinal velocity was chosen for this synchronisation, as it has the highest measurement accuracy common for both systems. The measurements from the Spacial dual are interpolated using the time-stamps of the Move-box. The Move-box will provide the inputs for the vehicle model so it is essential that these data points are not modified.

The tests manoeuvres were carried out multiple times to increase the significance of the measurements as experiment errors can be avoided. The vehicle is not perfectly symmetrical and may, therefore, affect the handling characteristics relative to the direction of the steering input. Other reasons for this can be attributed to differences in wear, operating history and bias of the components such as the steering system. To be able to analyse and compensate for this, all tests were done in either direction.

The raw experimental data contains noise and irregularities that may originate from the open loop test method or the irregular condition of the driving surface. These irregularities are filtered by a zero-phase second-order Butterworth filter which damps out frequencies above 2 Hz [12]. This is done without any large phase changes as the filter does not produce a large phase change at low frequencies.

The open loop nature of the experiments, the measurements must be scrutinized for abnormal behaviour that might produce big errors in the simulation, for example, due to the unevenness of the road surface. As the models used in this study are simplified, the data must roughly fit the assumptions made during model creation. As the tire model only considers pure lateral slip, the fluctuations of longitudinal velocity must be kept at a minimum to avoid inaccuracies of the model's results to occur. The datasets were also cropped to avoid low longitudinal velocity which affects the accuracy of the model, especially the calculation of the slip as shown in (2-1a).

Figure 3-1: The test vehicle on the test site in Valkenburg

3-3-1 Steady-state manoeuvre

The steady-state test is an open loop test that can be used to investigate vehicle handling such as understeer characteristics and cornering stability. This test exists in three variations, each varying one of the three main parameters, path radius, vehicle velocity and steering angle. For this study, the path radius was kept constant and the longitudinal velocity was increased incrementally. This is illustrated in Figure 3-3 which shows the vehicles trajectory and increasing speed in the form of a colour change. The choice of this method is due to the limited testing area that was available. The test area containing the concrete slabs was chosen due to its square area, the irregularities of the slab borders outweighed the restriction of the area with asphalted concrete.

According to ISO standards, the path radius should be 100 m which was laid out by means of cones which helped the driver keep the path radius as constant as possible. The Move-Box was used to control the longitudinal velocity which was increased from a standstill at a rate of $0.07\,\mathrm{m/s^2}$. The test continued until the handling limits of the vehicle were reached which due to the weather conditions and the relatively small path radius, was about $60\,\mathrm{km\,h^{-1}}$ [13].

The Understeer characteristics of a vehicle describe the relationship between longitudinal velocity and steering angle during a steady-state corner. Understeer occurs when the rear tires exhibit a higher cornering stiffness than that of the front tires. Due to this discrepancy with increasing longitudinal velocity, the yaw moment forces don't produce the desired yaw rate and therefore extra steering is needed to achieve the same cornering radius. Over-steer is when the opposite is true and the front wheels have a larger cornering stiffness than the rear, this leads a larger resultant yaw-rate and a counter steering compensation is needed.

$$\delta_{steer} = (1 - \frac{m}{2l^2}\frac{l_f K_f - l_r K_r}{K_f K_r} v_x)\frac{l}{\rho_0} \tag{3-1}$$

Equation 3-1 gives the relationship as an equation where ρ_0 is the steady-state cornering radius for the manoeuvre. The main factor that determines if a vehicle exhibits under-steer

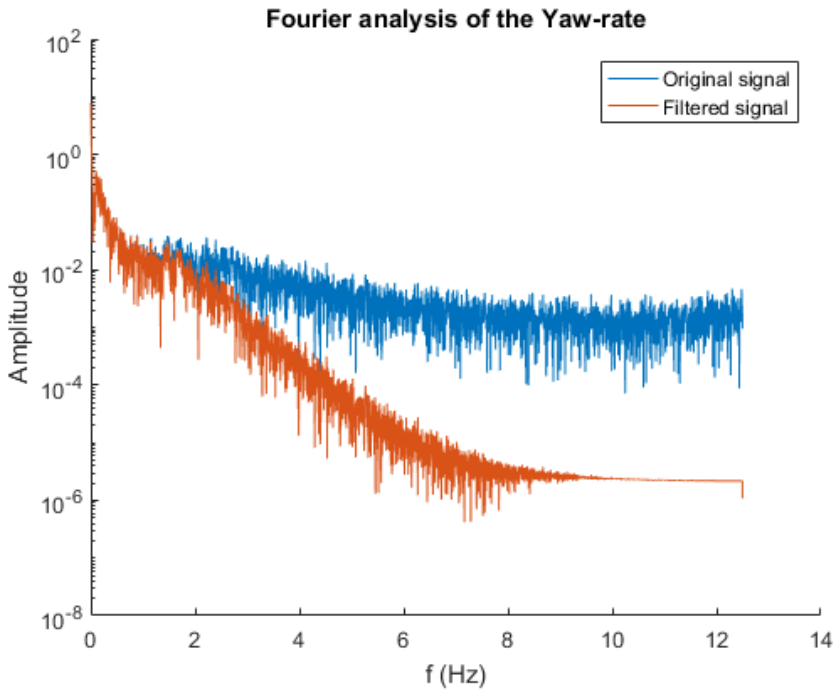


Figure 3-2: The yaw-rate freqency distribution before and after going through a second-order Butterworth filter

or over-steer is the value of $()l_fK_f - l_rK_r)$. Figure 3-4 shows that if $()l_fK_f - l_rK_r) < 0$ then the vehicle exhibits under-steer and steering needs to increased. If $()l_fK_f - l_rK_r) > 0$ over-steer occurs and the steering needs to be decreased. Lastly if $()l_fK_f - l_rK_r) = 0$ then the vehicle does not show any change in cornering radius when longitudinal velocity is increased. This characteristic is important as it can be used as a form of validation of the algorithm solution parameter sets.

3-3-2 Step manoeuvre

The step manoeuvre is a dynamic test that investigates vehicle handling behaviour during sudden changes in driver inputs. The test is carried out by accelerating the vehicle stand-still to a pre-defined longitudinal velocity of $60\,\mathrm{km\,h^{-1}}$ and kept constant. The heading of the vehicle is kept constant during acceleration and is turned manually after the required velocity has been obtained. The steering input is completed within 1 s to an amplitude large enough to obtain the desired lateral acceleration.

The resulting steering wheel angle is held constant as long as possible before being released to finish the test [14]. The original step response test procedure states that the steering angle should be held until a steady-state has been reached. However, this was not possible due to the limited testing area with the same road surface. The transition between the tarmac to the concrete slabs would interfere with the measurements and was therefore avoided.

This test differs from the steady-state test as it includes dynamic changes in the vehicles state and is, therefore, a better reflection of normal vehicle operating conditions. Vehicle modelling

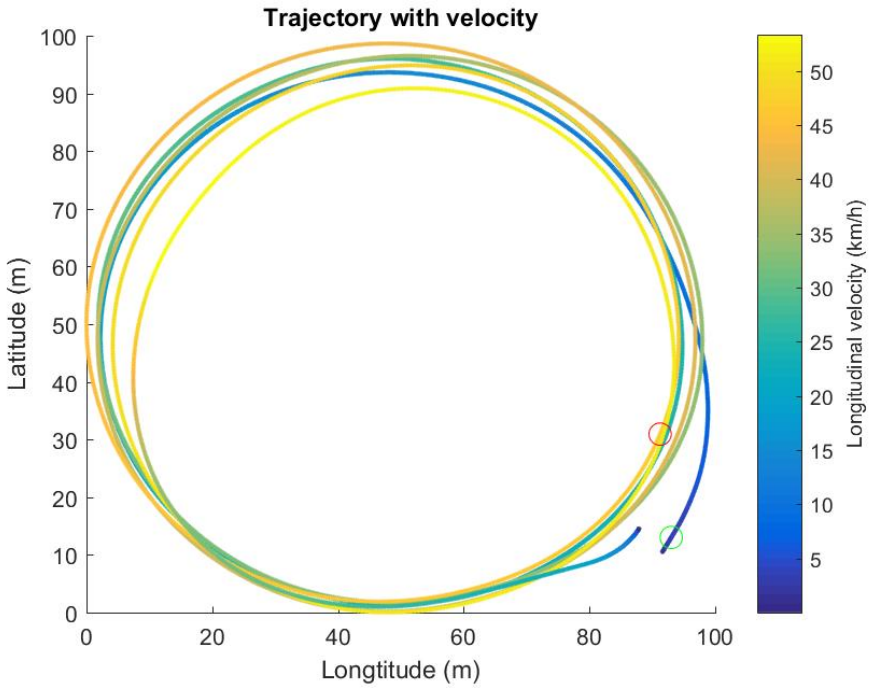


Figure 3-3: Plot of the vehicle trajectory during the steady-state test where the colours indicate the longitudinal velocity. The green circle marker indicates the start position and the red circle marker indicates the position where the test was ended due to loss of control

aspects such as component hysteresis and tire relaxation length are better reflected within the dynamics manoeuvres of this test. This is, therefore, a test that will provide a greater insight into the model optimization.

3-3-3 Validation data

The previous tests provide specific insight into how the test vehicle will behave in two different situations but does not reflect the driving conditions and manoeuvres of daily driving. Therefore a validation test set was recorded of the vehicle during one of the journeys between the university and the test site. The measurement equipment was initialised in the same way as the steady-state and step tests but was left on during the journey. The path of this journey is shown in figure 3-6 which shows the positional output of the GNSS and the longitudinal velocity recorded by the Move-Box. The dataset contains many driving operation modes including city, provincial and highway driving which is therefore separated into intervals limited by moments the vehicle's velocity was zero. Roads leading through areas with a lot of tall buildings were also avoided during the selection as this leads inaccuracies with the GNSS.

When compared to the testing datasets, the validation exceeds the scope of these tests. This is due to the larger velocity driven on the highway. As the optimization tests were performed at velocities averaging 40 to 60 km/h, intervals need to be chosen that will most likely to provide the best results. Figure B-8 and B-9 in the appendix show the intervals chosen that resemble the testing speeds of the optimization. Looking at the lateral acceleration values

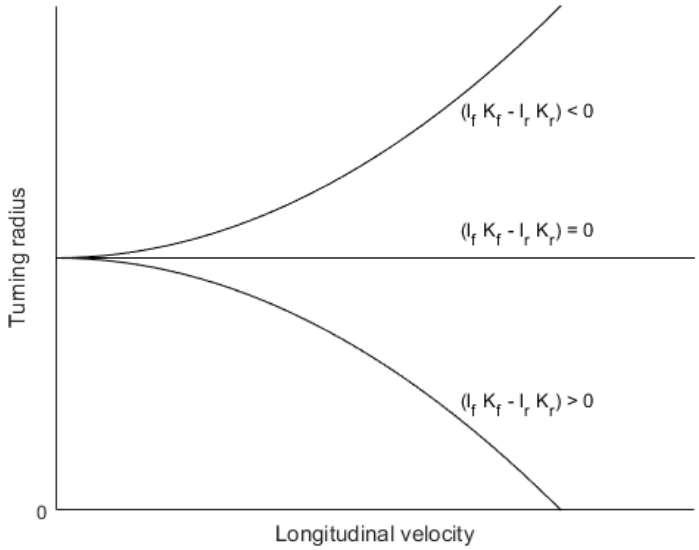


Figure 3-4: Turning radius change during increasing longitudinal velocity for different understeer characteristics [1]

in these tables, almost all of the manoeuvres seem to stay within the non-linear threshold of the tires. This does not mean that a non-linear model is not needed as the error accumulates over time due to the peaks of large lateral acceleration values, such as the maximum in the first dataset interval ($3\,\mathrm{m\,s^{-1}}$).

3-3-4 Discussion

The vehicle and the measurement systems provided the main limitations of the study, the biggest limitation being the absence of enough accurate information on the longitudinal behaviour of the vehicle. The experiment set up did, however, perform very well for lateral dynamics which is used extensively in this study.

The step and steady-state manoeuvre provide a diverse basis for a model to be optimized on due to its inclusion of dynamic and non-linear effects which may be observed on the open road. The ideal model must be able to simulate vehicle behaviour across a wide range of manoeuvre if it is required to reflect general operating conditions.

Different manoeuvres are influenced by different vehicle characteristics. For example, steering hysteresis is not a factor in a steady-state test due to the absence of oscillatory inputs while it is important when simulating a slalom manoeuvre. Even when it comes to the simulating for the same manoeuvre, every test dataset will have its unique uncertainties due to unseen changes in test conditions. Repetitions of the experimental tests are made to be able to catch these irregularities.

Each test was repeated three times in each direction but due to inaccuracies in the data, some datasets had to be left out of the overall data used by the optimization algorithms which leaves a total of 10 tests which are later used in the optimization of the vehicle model.

Figure 3-5: The path of the step manoeuvre on a satellite image of the test grounds

The multiple datasets also allow for cross-validation of the estimated parameters ensuring that the optimized vehicle model does indeed simulate the specific manoeuvre on which it was optimized as well as on the other test manoeuvre. Ultimately the dataset recording the return journey will be the final measure of how the model performs using the optimized parameters.

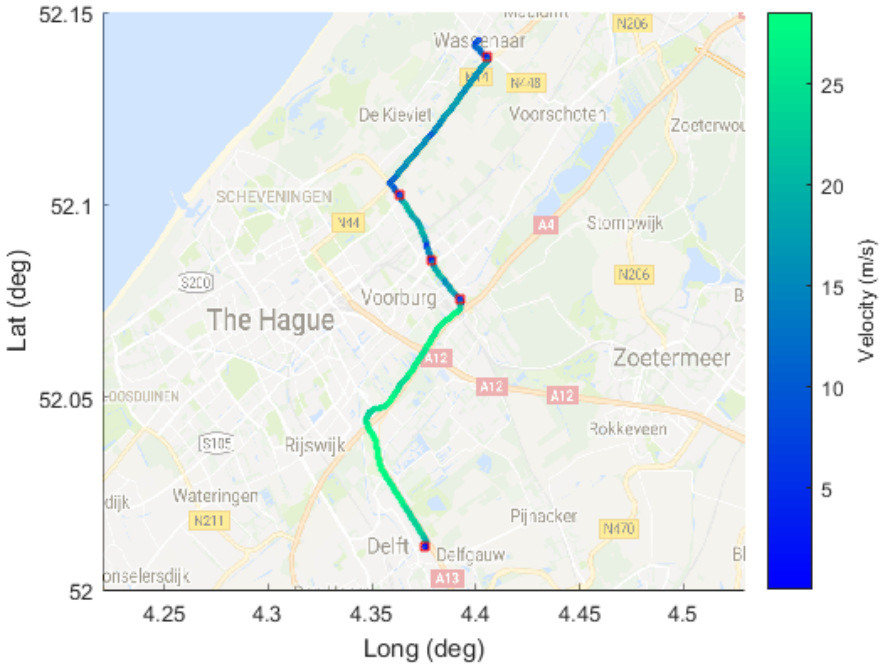


Figure 3-6: Trajectory plot of the return journey validation dataset where one part driven over a highway and the second part while within the destination city. The points marked with an O are moments when the vehicle was standing still.

Chapter 4

Model optimization

The most accurate method to solve an optimization problem is to perform a grid search. This evaluates every parameter value within the bounds to a certain resolution and is guaranteed to find the global minimum. A simple two-dimensional problem will not take much time to solve using a grid search. However, a multidimensional problem such estimating the Magic formula parameters will take a longer time to solve. This is due to the size of the search space which increases exponentially with the number of parameters to be optimized.

Optimization algorithms were created to decrease this search time by making use of the patterns and characteristics of the data. Not all points need to be calculated to find the global minimum and therefore trivial computations can be avoided. This also results in the biggest disadvantage being that the ultimate solution can be missed and the algorithm fooled to produce a lesser solution. Algorithms try to solve this problem using different methods based on the data characteristics while others use random processes to expand the search field and reduce the probability of premature convergence. Optimization algorithms can be categorised into two groups, Deterministic and Stochastic.

All algorithms require tuning of the internal parameters that determine how the algorithm will behave given the problem it is given to solve. Parameters for some algorithms, particularly stochastic algorithms, greatly affect its performance. Ideally, the parameters of the algorithms should be optimized themselves to ensure that the algorithm works at its best. The solution of this problem comes in the form of an adaptive algorithm which uses machine learning such as Temporal Difference Q Learning (TDQL).

This chapter will first cover aspects of model optimization such as the input parameter sets required and the calculation of the objective functions and how both objectives can be optimized. Both algorithms will be explained with respect to their theory and to the implementation within this study. This chapter will conclude with a method of standardization of termination criteria and a discussion their differences.

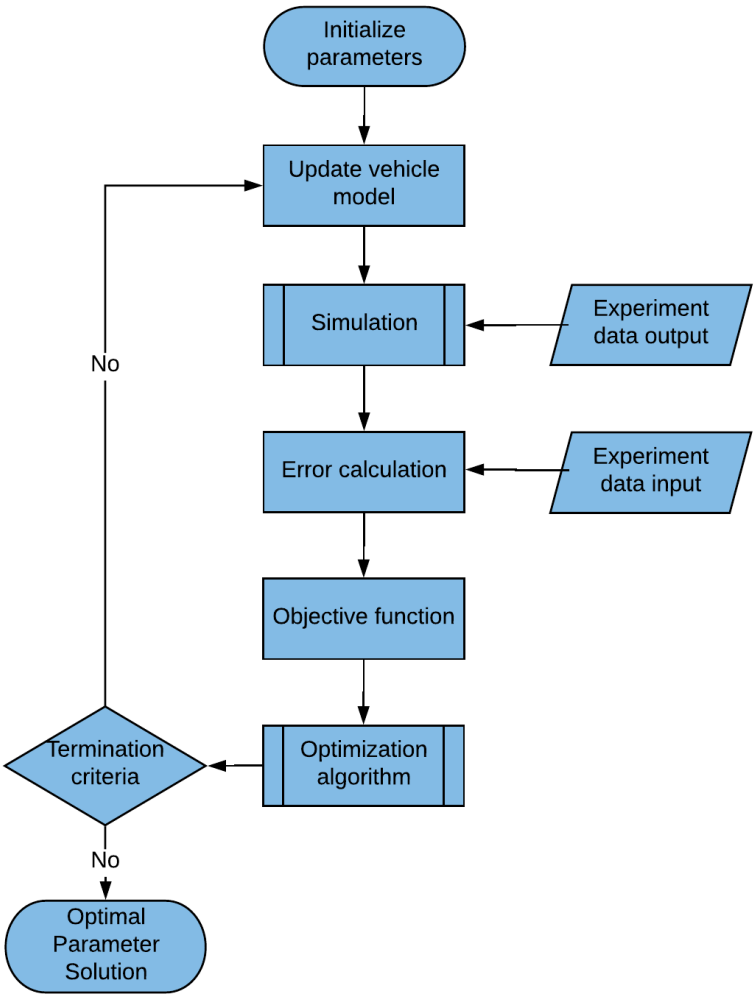


Figure 4-1: Flow chart for general parameter estimation

4-1 Parameter set

A parameter set is evaluated and the outcome is used to generate the next set of a parameter values for the next iteration. Local minima like troughs in the search space. They are able to fool algorithms into converging into an area it thinks contains global minimum. These points exist within the design space of vehicle models due to factors including model simplification, noise and measurement uncertainty.

All mechanical systems have constraints based on physical principles and laws. Models which include physical principles such as friction in the Magic formula must, therefore, have constraints to stop the optimization from providing impossible outcomes. Empirical models are bounded by the observations they are based on, which gives an indication of the range of parameter values that produce the best results. This range can also be set to force the algorithm to search in a narrower search field if a specific vehicle provides further indications that its parameters will not exceed a given range.

Bounds are the literal limits of a range of parameter values while constraints follow a specific equation that constrains the outcomes of the algorithm. The product of the Magic formula coefficients B, C, D produces the cornering stiffness at a slip angle equal to zero. This value can be constrained to a range of realistic values.

This constraint function is called an inequality function, as a calculated value must be more or less a given value. An equality constraint function constrains an output of an algorithm to a specific value but does not appear in this study. To simplify the algorithms, only bounds will be implemented while constraints are recommended for further research.

Table 4-1: Bounds and estimated initial parameter set

Parameters	Upper	Initial	Lower
DF	0.65	0.75	0.95
CF	1.15	1.56	1.7
BF	5	17.09	20
EF	-1	-0.88	1
SvF	-0.01	0	0.01
ShF	-0.005	0	0.005
DR	0.65	0.75	0.95
CR	1.15	1.56	1.7
BR	5	17.09	20
ER	-1	-0.88	1
SvR	-0.01	0	0.01
ShR	-0.005	0	0.005
RLF	0.1	0.35	0.5
RLR	0.1	0.27	0.5

Parameter values of past tire experiments are used to make a rough estimate of the initial parameter set and the reasonable bounds for these parameters. These values are presented in table 4-1. These parameters are primarily used by the deterministic algorithms. Estimates were made with the help of tire files with a similar tire type and dimension [15]. Parameters such as the friction coefficient (DF and DR) were reduced due to the wet conditions prevalent

during the measurement experiments. The values for relaxation length were determined as estimates based on various literature [1, 9].

4-2 Objective function

Before a model has been optimized, the outputs of the model will not resemble the measured data as there will be errors in the model parameters that produce inaccuracies in the simulation output. Reducing this error is the goal of the optimization algorithm and therefore this error must be quantitatively defined and is called an objective function.

There are many methods to do this but most are based on the absolute difference between corresponding points on both datasets but there are also methods that use other statistical methods such as the comparison of standard deviation.

$$R_i(\mathbf{x}) = y_i - f_i(\mathbf{x}) \tag{4-1a}$$

$$S(x) = (R_i(\mathbf{x}))^2 \tag{4-1b}$$

Equation (4-1a) refers the to the residual $R_i(\mathbf{x})$ at data point i where y_i is the experimental data output and $f(x_i)$ is the model output at point i. Equation 4-1b is called the residuals squared $S(\mathbf{x})$ which is an dimensionless value that is used by many deterministic algorithms that apply least squares method.

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n} S_i(x)}{n}} \tag{4-2}$$

Equation 4-2 shows the equation for the Root Mean Squared Deviation (RMSD). A modification of $RMS(x)$ to allow for comparison between data sets of different length. It does this by calculating the average of the error over the number of data points. This equation produces a function value with the same units as the data recorded and being simulated. This is good for the understanding of the algorithm and model outputs but due to the multi-objective nature of this optimization, comparing objective functions is even more useful.

While objective functions can be normalized during the process of the optimization, it is easier to normalize the objective function values within the fitness function. To do this, the RMSD is divided by the average of the experimental data values,$\hat{y}_i$, for that dataset as seen in equation 4-3. This is called the Normalized Root Mean Squared Deviation (NRMSD).

$$NRMSD = \frac{RMSD}{\hat{y}_i} \tag{4-3}$$

4-3 Multiple objectives

As explained in chapter 2, the vehicle model will be optimized for the yaw rate and lateral acceleration. These values describe the rotational and translational motion of the vehicle and are the focus of the equations of motion of the vehicle model. The error between the model output and the measured data should be minimized for both motions which describes a multi-objective optimization. The output of these optimization problem consists of multiple solutions called a Pareto front. They span between the optimal solution for either motion as the decrease in strength of one objective is compensated by the other.

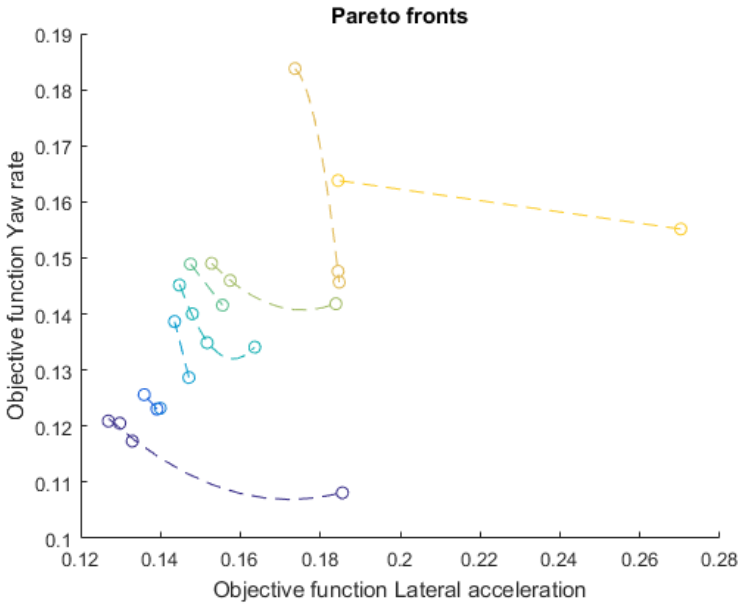


Figure 4-2: Illustration of the Pareto fronts that exist within a objective search space

Figure 4-2 shows a number of Pareto fronts at intermediary points during the progress of a multi-objective algorithm. The determination of which front an individual belongs to is made by a process of elimination called non-dominated sorting which will be explained in detail in section 4-5-2. Each front consists of solutions that are better than that of the previous front although it is possible that some solutions remain the same and that the front is only made more accurate. The objective is to minimize both objectives until the front lies as close to the origin of the graph as possible.

To find the solution that is the most suitable for the vehicle model a decision must be made which will be chosen as the final solution. While this decision is made internally when it comes to deterministic algorithms, stochastic algorithms produce a set of solutions with a large Pareto front where a solution can be manually chosen. The advantages of a Pareto front is that one can make an informed decision depending on the various solutions produced by the algorithm [16].

For this thesis, both objectives are just as desirable and therefore an even solution will be chosen from the resulting Pareto front. The method used is to calculate the Euclidean distance of the objective function values of all the Pareto solutions to zero and choose the solution with the least distance.

4-4 Deterministic algorithms

Deterministic algorithms use analytical properties such as the position, gradient and objective function values to approximate the characteristics of the search space and move in the direction of a minimum solution. As the name suggests, deterministic algorithms produce the same results when run using identical data and starting positions. This type of algorithm will most likely converge towards a local solution as it only obtains information about its immediate surroundings and the data of previous iterations. Automotive engineers frequently use this algorithm to tune model parameters based on knowledge of the tire behaviour and available tire information. While this method might work well for optimization using tire forces, body inertial data consists of more noise and uncertainty which may increase the prevalence of local solutions in the area of the initial estimate.

Deterministic algorithms used in this thesis will be implemented using the algorithm functions available in Matlab. The algorithms used by these functions are largely based on variations of the Newton-Gauss method which is explained in appendix A-2. Specific information regarding the functions used was found in the Matlab documentation of the Optimization toolbox [17]. Clarification was sought in papers documenting the developments of these algorithms [18].

First, the Trust-region-reflective algorithm is explained as this is the algorithm used in the *lsqnonlin* Matlab function that will be used in this thesis. This is coupled with the Matlab goal attainment algorithm *fgoalattain* to produce a multi-objective solution to both the lateral acceleration and the yaw-rate.

4-4-1 Trust-region-reflective algorithm

The Trust-region-reflective algorithm is a constrained non-linear optimization algorithm that is based on the Newton method and the principle of trust regions. A solution is found by following iterative steps to approximate the locally bound search space surrounding the current point in the iteration.

$$S(\mathbf{X}_k + \mathbf{d}) \approx M(\mathbf{d}) = S(\mathbf{X}_k) + \Delta\mathcal{F}(\mathbf{X}_k)^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\mathbf{H}_k\mathbf{d} \tag{4-4}$$

The approximation of the local search space is done by means of a second order Taylor series as shown in equation 4-4. Where the function $M(\mathbf{d})$ approximates the residual squared, $S(\mathbf{X})$ using $\mathbf{X}_k$ as the parameter set. $\Delta S(\mathbf{X})$ and $\mathbf{H}$ represent the gradient and Hessian at $\mathbf{X}_k$ during iteration k.

$$\begin{aligned} \underset{d}{\text{minimize}} \quad & M_t(\mathbf{d}) \\ \text{subject to} \quad & ||\mathbf{d}_k|| \leq \Delta_k \end{aligned} \tag{4-5}$$

The goal of this algorithm is to determine the Newton step $\mathbf{d}_k$ by solving the minimization problem 4-5. This problem is subjected to conditions where $||\mathbf{d}_k||$ is the norm of the newton step and Δ_k is a scalar radius that defines the trust region.

$$\Delta M(\mathbf{d}_k) = \Delta S(\mathbf{X}_k) + \mathbf{H}_k \mathbf{d}_k = 0 \tag{4-6}$$

$$\mathbf{d}_k = -\mathbf{H}_k^T \Delta S(\mathbf{X}_k) \tag{4-7}$$

Differentiating equation 4-4 with respect to $\mathbf{d}_k$ produces equation 4-6 which is solved to obtain $\mathbf{d}_k$. Consequently the new $\mathbf{X}_{k+1}$ can be calculated by adding this step to the solution of the previous iteration, $\mathbf{X}_{k+1} = \mathbf{X}_k + \mathbf{d}_k$ [20]. If the new $\mathbf{X}_{k+1} < \mathbf{X}_k$ the new parameter set is accepted and the next iteration is begun. If this is not the case, the trust region radius is decreased and the iteration is restarted [19].

The finite difference method is used to calculate an approximation of the gradient by varying each parameter and recording the difference of the function value for the optimization problem, in this case the residual squared $S(\mathbf{X}_k)$. The Hessian matrix a matrix containing the second order partial derivatives of the problem function with respect to the current parameter set. While this too can be calculated using the finite difference method, this produces a poor quality approximation and requires a lot of computational power. In general, obtaining an accurate Hessian matrix requires large computational power and therefore is usually approximated using data from previous iterations. A Hessian update method is a positive definite quasi-Newton approximation called the Broyden Fletcher Goldfarb Shanno (BFGS) method which updates the Hessian matrix after every iteration.

For a Newton method optimization to work, the Hessian matrix has to be positive definite, which means that the approximated curvature of the search space must be positive. The trust-region method goes about this by changing the trust region to accommodate indefinite Hessian matrices and will try to make a large step to move into an area with better chance of continuing . Another method to avoid not converging to a solution is called the Levenberg-Marquardt method which changes the Hessian until it is feasible or until it forms close to a identity matrix. This means that the algorithm is transformed into a steepest descent algorithm [20].

When the algorithm reaches a boundary set by the user, it will try to change the direction of the merit function to reflect and move back towards the centre of the search space. This is why the algorithm is called the Trust-region-reflective algorithm.

4-4-2 Goal attainment algorithm

Most algorithms cater optimization problems with a single objective. To accommodate multiple objectives modifications have been made to existing algorithms to make this possible. The biggest challenge to multi-objective optimization is determining the priority of the objectives to produce a global solution. One of the Matlab functions that do this is the *fminimax* function which prioritizes the weaker of the two objectives during each iteration. While this method works well with normalized objective values, the possible solution of either objective might differ. The overall progress might be hindered due to an objective given priority, while no further improvements are possible.

fgoalattain is another multi-objective optimization function but differs as it does not aim to minimize the objectives to zero but to a pre-determined set of goals. These goals are not

known for this problem and therefore must first be determined by independent single objective functions such as *lsqnonlin*, which is based on a non-linear least squares method using the Trust Region algorithm explained in the previous section.

This method of goal attainment gives the algorithm more information to prioritize the objective that is furthest from its optimum solution. If one or both objectives have reached their goals, the algorithm does not stop but will try to minimize the objectives further until one of the termination criteria has been reached. This ensures that if the single objective algorithms produce a false solution, the algorithm will continue until a better solution has been found. These situations give further reason to distrust the initial parameter set and prove the faults of deterministic algorithms.

The Matlab *fgoalattain* function uses a variation of an Sequential Quadratic Programming (SQP) algorithm, which is the steepest descent method using the gradient of a local search field to determine the search direction and the next iteration point. To allow a multi-objective optimization, the objective function needs to be modified to accept both objective function values for lateral acceleration and yaw rate.

$$\underset{x}{\text{minimize}} \quad f = \max_i \frac{\mathcal{F}_i(\mathbf{X}) - \mathcal{F}_i^*}{w_i} \tag{4-8}$$

The new problem formulation is shown in 4-8 where $\mathcal{F}_i^*$ is the objective goals and w_i is the weight of the specific objective. This term determines how strict the algorithms consider the goals. With a w_i more than 1, the algorithm aims to overachieve these goals as previously mentioned while a w_i of 1 aims to reach the goals precisely.

This goal attain problem formulation is a modification by Brayton et al. [21] of a previous formulation which uses the separate objective functions as constraints. These constraints were formulated in a way that the separate objective functions had to lie within their respective goals for the solution to be feasible. This modification was made because it is difficult to define which constraint is more important relative to other constraints such as the bounds.

Figure 4-3 shows a flowchart that illustrates the steps taken during the algorithm iteration. The algorithm is initialized with the initial parameter set provided in the previous section and a positive definite initial Hessian matrix. The parameter set is first evaluated and the weakest objective value determines which objective will be used in the current iteration.

$$\mathcal{L}(\mathbf{X}, \boldsymbol{\lambda}) = f(\mathbf{X}) - \boldsymbol{\lambda}^T \mathbf{B}(\mathbf{X}) \tag{4-9}$$

The ensure that the model parameters do not exceed the bounds, a constraint equation needs to be taken into account. The resulting equation is called a Lagrangian equation $\mathcal{L}$ and is shown by (4-9) where $\mathcal{B}$ is the matrix containing the bounds and $\boldsymbol{\lambda}$ is the Lagrangian Karush Kuhn Tucker (KKT) multiplier for the bounds. This multiplier indicates how active a constraint influences the Lagrangian equation. Which means that $\boldsymbol{\lambda} > 0$ indicates that the constraint is strongly active while $\boldsymbol{\lambda} < 0$ indicates that the constraint is inactive and can be disregarded [22].

$$\begin{aligned} \underset{d}{\text{minimize}} \quad & \frac{1}{2}(\mathbf{d}^T \mathbf{H} \mathbf{d}) + \Delta f^T \mathbf{d} \\ \text{subject to} \quad & \mathbf{B}(\mathbf{X}) + \Delta \mathbf{B}(\mathbf{X})^T \mathbf{d} \leq 0 \end{aligned} \tag{4-10}$$

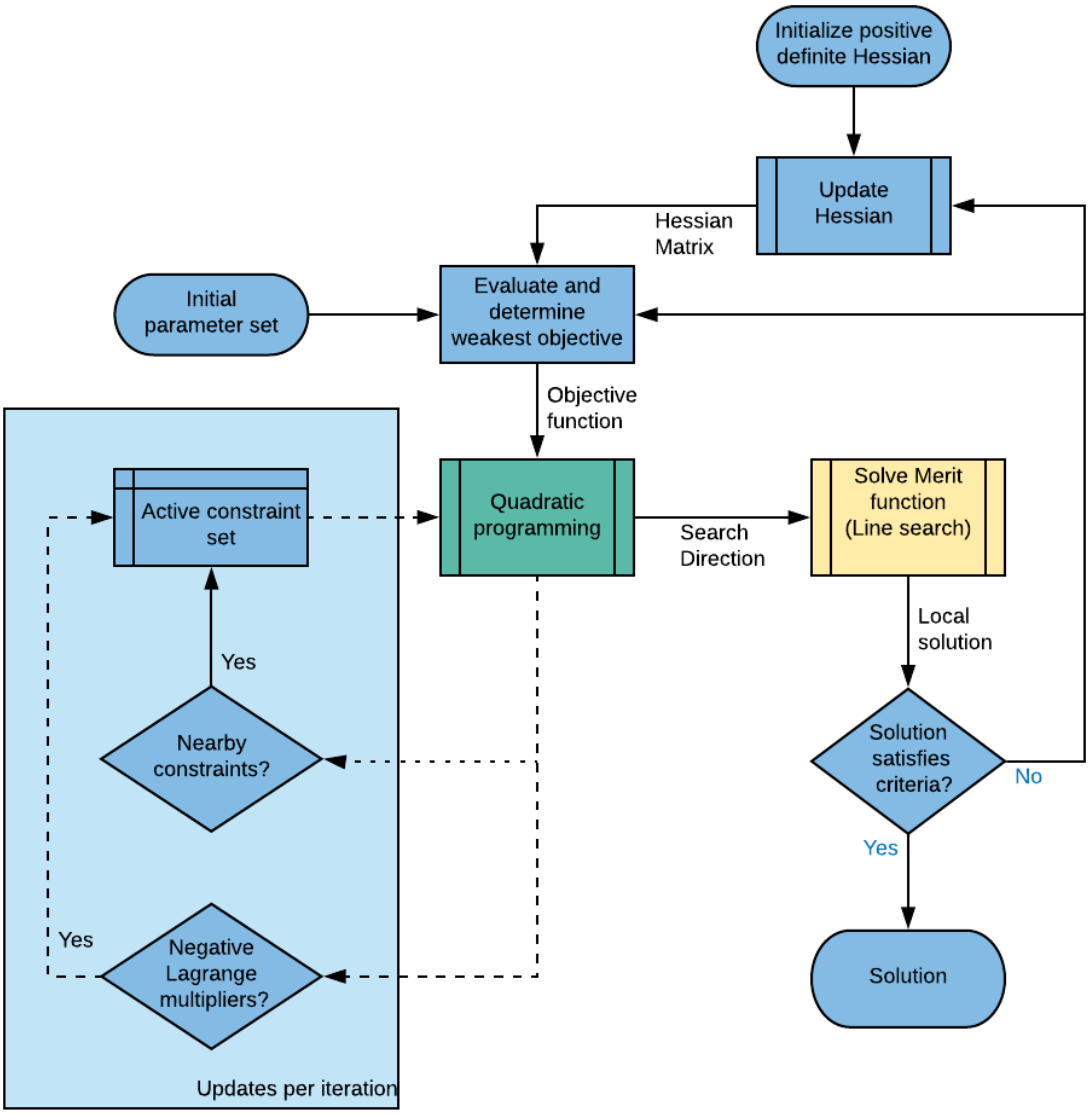


Figure 4-3: Flow chart of the Sequential Quadratic Programming algorithm used in *fgoalattain*

The search vector $\mathbf{d}$ is minimized by local quadratic programming. The differentiation of the Lagrangian equation produces 4-10 which is the local quadratic sub-problem to be solved. This problem is solved in the same manner as the Newton-Gauss method explained in section A-2.

As mentioned before, the bounds can be either active or inactive depending on the location of the search. The algorithm takes this into account by checking to see if the bounds are within a unity distance of the search direction at every iteration. If this is the case the respective bound is added to $\mathbf{B}(\mathbf{X})$ and is considered active during the next iteration. If in the next iteration the KKT multiplier for that bound becomes negative, this means that the bound is

no longer active and it removed from $\mathbf{B}(\mathbf{X})$.

$$\mathbf{X}_{k+1} = \mathbf{X}_k + \delta_{algo}\mathbf{d} \tag{4-11a}$$

$$\Psi(\mathbf{X}) = \max_i \frac{\mathcal{F}_i(\mathbf{X}) - \mathcal{F}_i^*}{w_i} \tag{4-11b}$$

The merit function, shown in equation 4-11b, is minimized by a line search, shown in equation (4-11a), in the direction d which was obtained during the quadratic program. The algorithm continues its iteration until one of the termination criteria have been met. This may include the following, depending on the threshold set before the algorithm is initialized.

- The change in objective function value across an iteration drops below a certain value indicating that a bottom of a trough has been found.
- The step value of the iteration or the sub-problem of the algorithm has decreased below a certain value indicating that the approximated search area is too small to provide large benefit for the objective function value.
- The number of iterations exceed a predefined value to ensure that the algorithm is stopped when the algorithm fails or takes a very long time to converge to a solution.

There is also a criterion that evaluates the optimality of the objective function output. This is the gradient of the output and assumes that this reduces to zero when an optimal solution has been found. This criterion is not used in the name of comparison as there is no equivalent for stochastic algorithms.

4-5 Stochastic algorithms

Stochastic algorithms are based on evaluating random parameter sets and basis its search only on the objective function value of the problem. This makes stochastic algorithms simplistic and applicable to many problems of all development disciplines as it does not need prior knowledge. One of the biggest differences with deterministic methods is that it considers a population of many individual points within the parameter bounds, also called the design space. These points are evaluated and the resulting fitness function is used to determine the population of the next iteration. The number and distribution of these points at the start of the optimization is chosen by the user. This search method has larger parameter sets as it needs many more points to get a proper representation of the search field. This can lead to a disadvantage of having a longer computation time per iteration as all population points need to be evaluated separately. Many of these algorithms are inspired by biological behaviours such as natural selection and swarm theory. The basic knowledge of Genetic algorithms was retrieved from books such as from *Introduction to Genetic algorithms* by Deepa et al. [23] which provide a wide introduction to the topic of stochastic algorithms as well as a guide to searching for other relevant literature.

4-5-1 Genetic algorithms

The genetic algorithm is based on biological observation of the evolution of a population considering the change in individuals during pairing and mutation. A parameter set is considered an individual and the parameters seen as the genes of this individual. A fixed population of individuals are uniformly generated at random across the search field. The individuals are then evaluated to produce their objective value before being sent to algorithm operators to modify and change them for the next generation. An operator of a genetic algorithm can be programmed to make any change to an individual or a pair of individuals as long as the basic principles of the algorithm are upheld. The probability of an individual to be chosen for this process depends on their fitness function and a set algorithm parameters set by the user.

The standard genetic algorithm use the following operators:

- A random selection is made for reproduction based on the fitness value of the individuals.
- Pairs are randomly chosen and are mated together to produce offspring. This operator is known as a crossover as the offspring is made up of a cross of different genes.
- A mutation operator applies random deformation on the offspring to ensure that the algorithm does not prematurely converge to a local minimum and increases the chance of exploring other area's of the search field.
- The new offspring are then evaluated to produce their fitness value.
- The last step is the placement of the offspring in the population which has entered into a new generation. The offspring is compared with parent that fathered it and replaces if it has a better fitness value.

This process is repeated until the solution of an iteration is determined to be the global solution to a certain accuracy. As mentioned before, determining when a solution has been reached is one of the biggest challenges for an algorithm as there are still many different parameter combinations to check. Therefore termination criteria have been defined to help determine this. For a stochastic algorithm, these are based on the current performance of the algorithm as well as the progress of the objective function values of the population. Most termination criteria work on the principle that the algorithm is terminated when a certain aspect of the individuals drop below a pre-defined threshold. Some examples of these aspects are:

- The change in objective function value of the best individual which indicates that the algorithm can not find a better solution to replace the current best individual.
- The average change in objective function value over a consecutive number of generations which gives the algorithm a chance to further search for a better alternative.
- The average of the entire population which gives an indication of the algorithms overall progress.
- The objective function value of the worst individual to ensure that the algorithm has achieved a minimum requirement and will only produce a better result.

- A sum of all the individuals will give an overall measure of the the optimum area and act as an criteria to stop the algorithm. However decisions need to be made if all the individuals are sufficient enough to be considered [23].

The threshold for these termination criteria is difficult to determine as some require an absolute objective function value which requires knowledge of the potential of the algorithm and the results of the vehicle model. Relative criteria such as the average change in objective value are easier to define but may cause the algorithm to continue searching which increases the computational time unnecessarily.

Simpler forms of termination criteria include terminating the algorithm when a maximum number of generations have passed. This criterion is mostly used when specifically comparing algorithms with each other as it gives a good indication of the algorithm's performance. This criterion is used in this study as a way to ensure that algorithm runs that do not show the required performance are stopped before too much computational power has been spent.

Multiple stopping criteria can be used to complement each other in a way that the overall results of the algorithm are op to the required standard. For example, the main termination criteria may be based on the rate of change of the main individual but can only be triggered once all other individuals or the average of these individuals have passed a certain threshold which gives certainty to the final solution.

4-5-2 Adaptive Algorithm: Demo-TDQL algorithm

The stochastic genetic algorithms work on the principle of exploring or exploiting the search space. A large spread of test points, or individuals, is beneficial as these maybe reside in areas which might contain the global minimum. Exploitation generally means that this chance is high and that the algorithm must investigate the area further to determine if it truly does contain the global minimum. During this time, exploration of other areas must also continue in the case that this area is not what it seems. The two methods work against each other as there is a fixed number of available individuals and therefore difficult to find a balance to produce the best result.

The parameters such as the population size and the operator probabilities influence these characteristics and therefore need to be chosen wisely to fulfil the needs of the data. For a generic algorithm, these parameters are set and kept constant throughout the optimization. Trial and error are used to determine which parameters perform the best while as explained above, the effectiveness depends on the area that is being searched.

There are many methods to adapt this parameter during the optimization such one proposed by Deepa et al. [23] which works by changing the mutation operator relative to the algorithms rate of progress. The mutation probability is increased when minimal progress is made, this improves the exploration of other areas. While if the population improves by large amounts, it is seen as good path to exploit and the mutation probability is decreased.

A trial and error logic approach to this problem is proposed by Ortiz et al [24]. This algorithm produces two populations per generation using different algorithm parameters. The performance of these populations determines which parameter value will be used for the next generation. This method is feasible for small datasets such as that of laboratory tire force

model optimization in Ortiz et al but is not feasible for this study as the simulations are much bigger. Producing two populations per generation will double the amount of time the algorithm needs to converge to a solution.

Machine learning is a method that acts as a separate function that observes the state and actions of the algorithm. This data is recorded by the function and a reward given to the algorithm which influences its characteristics. The goal of machine learning is to determine the best parameter policy to produce the highest return based on its interaction with algorithm [25]. This adaptive function does not need prior knowledge of the system but gains knowledge at every generation on how to improve the results.

A machine learning algorithm called Differential evolution for Multi-objective optimization (DEMO) TDQL was studied as a genetic algorithm that uses machine learning to change the probability of the mutation operator. The state transmitted to the machine learning function is the population and its rank according to its fitness, the action to be monitors is the outcome of a certain probability value for that generation and the reward is the chance of that probability value being chosen again for that rank in the population. All this information is stored in a Q-table which is updated after every generation.

This algorithm is described by Rakshit et al. [26] and will covered in the following sections as well used as the adaptive genetic algorithm to answer the research question of this thesis. Figure 4-4 is a flowchart that outlines the steps taken by the algorithm.

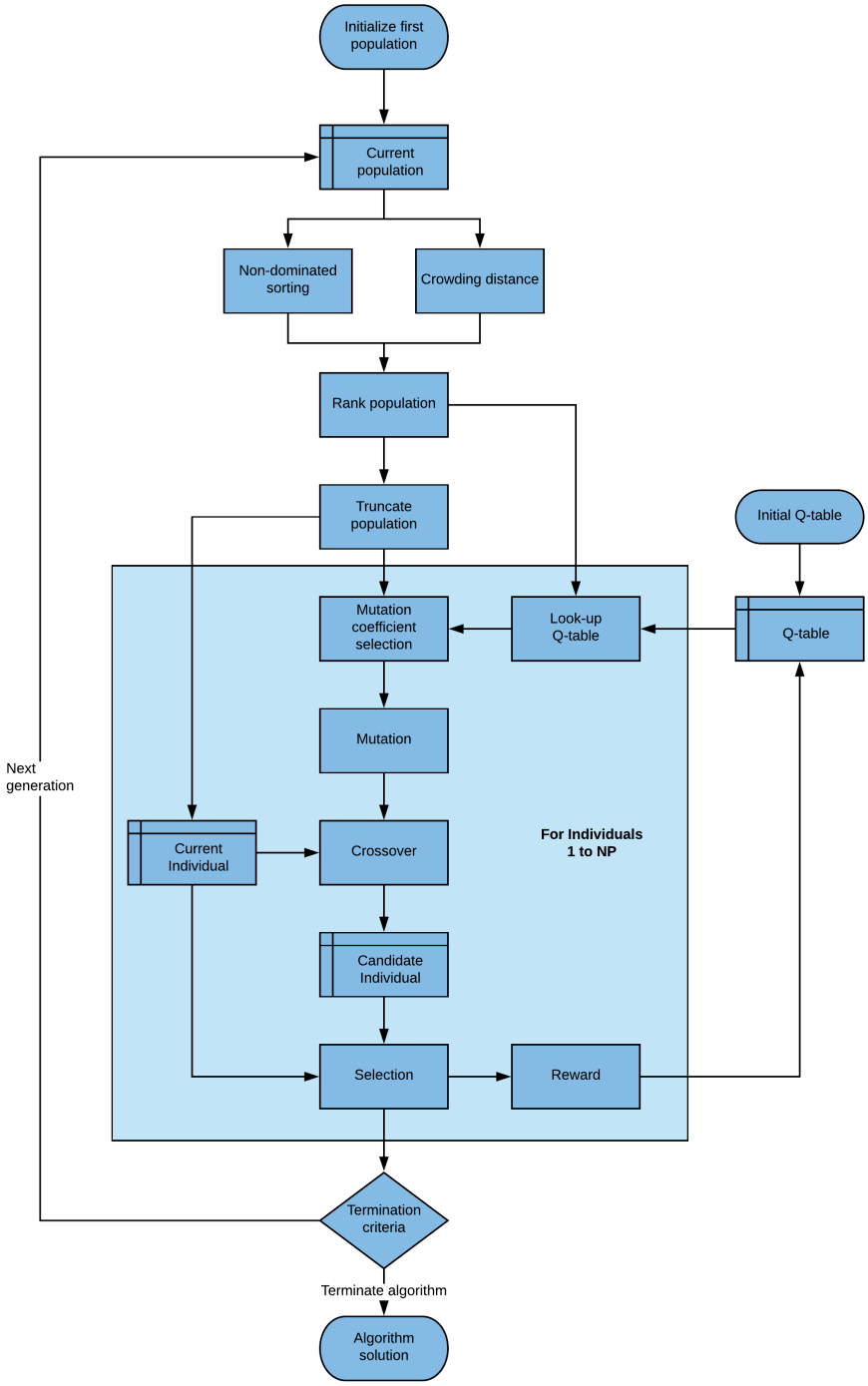


Figure 4-4: Flowchart of the DEMO TDQL algorithm

Non-dominated sorting

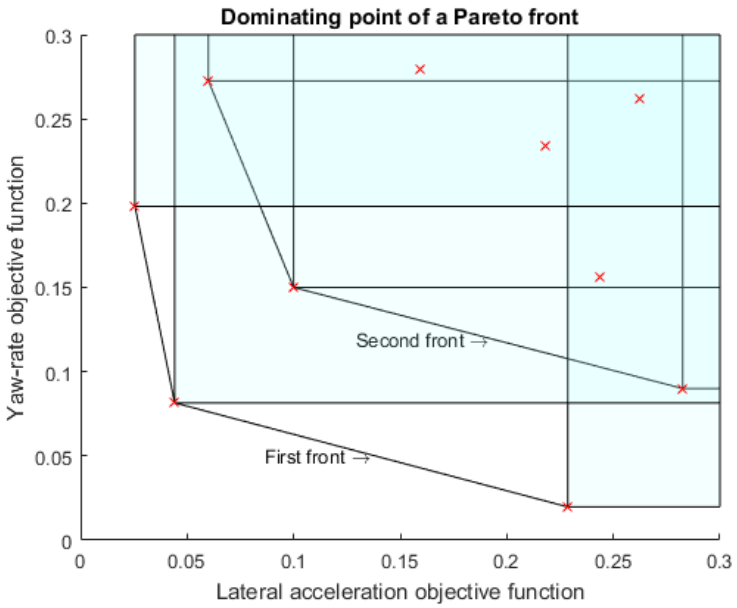


Figure 4-5: Dominating points and their fronts

The optimal solutions of a multi-objective optimization belong to a Pareto front, explained in 4-3. Determining which individual solutions belong to the optimal front is call non-dominating sorting. The solutions behind the optimal front also belong to Pareto fronts and can be considered as a non-dominated front when members of the previous dominating fronts are not considered. This can be done for all solutions and determines which Pareto front PF_i a solution belongs to. This is illustrated in figure 4-5, which show the first front, otherwise known as the optimal front, and the solutions that are dominated per solution in this front. The second front also dominates a set of solutions per front solution and this continues until all fronts have been identified [26].

One solution is said to dominate the other solution when two criteria has been met:

- The solution is not worse than the other in all objective function values.
- One of the function values of the solution must be strictly better than the same value of the other solution.

Crowding distance

The crowding distance of the individual solution is a measure of how unique this solution is within a Pareto front. If two solutions are situated close together in a Pareto front, one of them has a reduced significance when it comes to defining the Pareto front. This is definitely true for the solutions at the extreme ends of the Pareto front, if these are close together, a front is difficult to define outside of these two solutions. A Pareto front is properly defined if

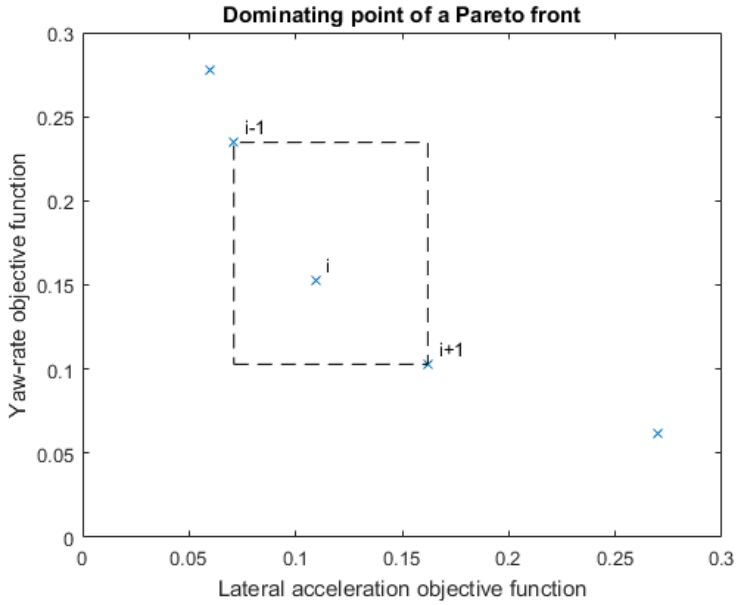


Figure 4-6: Illustration of the method of calculating the crowding distance

its solutions are located a large distance from each other. To achieve this, solutions with a large crowding distance are given a higher rank and are given priority.

$$CD_i = \begin{cases} \dfrac{\mathcal{F}(i+1)_{la} - \mathcal{F}(i-1)_{la}}{\mathcal{F}_{la}^{max} - \mathcal{F}_{la}^{min}} + \dfrac{\mathcal{F}(i+1)_{yr} - \mathcal{F}(i-1)_{yr}}{\mathcal{F}_{yr}^{max} - \mathcal{F}_{yr}^{min}} & \text{for } 1 < i < end \\ \infty & \text{otherwise} \end{cases} \tag{4-12}$$

The crowding distance CD_i is calculated in equation (4-12) where i represents the solution number in the Pareto front and The neighbours of the solution in question are $\mathcal{F}(i+1)_m$ and $\mathcal{F}(i-1)_m$ which together form a rectangle. The distance is normalised using $\mathcal{F}_m^{max}$ and $\mathcal{F}_m^{min}$ with $m \in la, yr$ which represent the maximum and minimum objective function value found in that Pareto front.

The solutions that lie on the boundary of the Pareto front are considered to have an infinite crowding distance and accordingly given the highest priority. If multiple situations arise where two or more solutions hold the same fitness values, this method of crowding distance can lead to instability due to the fact that many distances will be zero and do not productively take part in the algorithm. After many algorithm runs, it is observed that this situation does not arise frequently enough to produce this instability [27].

Ranking

At this point, every individual in the generation population has a Pareto front and a crowding distance. A final rank can be given to each individual by sorting them by means of a ranking policy which determines that individuals importance. In this case, the individuals are sorted in ascending order depending on their value of $PF_i(t)/CD_i(t) \forall i$. This means that individuals

with low crowding distance will be given a lower rank while an individual with a larger crowding distance might be given a higher rank despite being in a lower Pareto front. The position of the individuals in the population does not change, but each individual's rank is listed in a separate vector $\mathbf{R}_i$ [26].

Mutation operator and Mutation factor selection

The purpose of the mutation operator is to produce an alternative individual $\mathbf{V}$ within the search boundaries to later be evaluated by the algorithm and is the biggest factor of the algorithms search behaviour. There are many mutation operators used by the literature and differ from each other by their search characteristics. Some mutation operators are more exploitative and focus their search near the last best results while other operators tend to focus more on exploring the search area. Every operator has the option to tune these characteristics with mutation coefficient [28]. Random difference vectors, which are created by taking the difference between two random individuals chosen from the current population.

$$\mathbf{V}_i(t) = \mathbf{X}(t)_i + F_1'(\mathbf{X}_{best}(t) - \mathbf{X}_i(t)) + F_2'(\mathbf{X}_{rand1}(t) - \mathbf{X}_{rand2}(t)) \tag{4-13}$$

Equation 4-13 shows the operator that was chosen for this study and is used by the paper of this algorithm. It takes a point in between the current individual $\mathbf{X}(t)_i$ and a random individual taken from the best Pareto front $\mathbf{X}_{best}(t)$ which is due to the fact that all the individuals in this front are considered to be the best solution. A random difference vector consisting of two randomly chosen individual from the population, $\mathbf{X}_{rand1}(t)$ and $\mathbf{X}_{rand2}(t)$, is added to ensure that the operator does not only search towards the best individual.

Two mutation coefficients, F_1' and F_2', control how the operator mutates the individual. This means that the algorithm requires two Q-tables to adaptively change each coefficient. However to simplify the operator for this study the two mutation factors are set equal to each other to reduce the number of Q-tables [26].

$$Q(t) = \begin{array}{c} \\ R_i \downarrow \end{array} \begin{array}{c} \xrightarrow{\qquad\qquad F_j \qquad\qquad} \\ \begin{array}{cccccccccc} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1 \end{array} \end{array} \begin{array}{c} \\ \begin{array}{c} 1 \\ 1 \\ 2 \\ 3 \\ \vdots \\ NP \end{array} \end{array} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{4-14}$$

$$P(F_j) = \frac{Q(R_i(t), j)}{\sum_{k=1}^{10} Q(R_i(t), k)} \tag{4-15}$$

Equation 4-14 shows the newly initialized Q-table where columns signify the range of optimization parameters, in this case, the mutation coefficient, and the rows signify the rank of an individual R_i. The mutation factor F_j is chosen at random out of a range of 0.1 to 1.

The probability of a mutation coefficient being chosen is determined by its respective Q-table value for the rank of the individual and is calculated with equation 4-15. $Q(R_1(t), 8)$ is shown in equation (4-14) is encircled in red and $\sum_{k=1}^{10} Q(R_1(t), k)$ is the sum of the values encircled in black.

The parameter that results from the operator may lie outside of the bounds and therefore needs to be rectified. There are many methods to do this, the simple method is to discard the result and repeat the operator until a valid parameter set is found. This particular operator has a rejection rate of 24 % which is considered high but as the computational power needed to run the operator is very low and thus repeating the operator does not reduce the overall performance of the algorithm. A penalty could also be added to the respective mutation coefficient in the Q-table to a reduced probability of this error happening again but as larger mutation factors have a higher chance of mutating the individual beyond the bounds, this would skew the Q-matrix towards smaller mutation factors. Another method put forth by [24] is to change the invalid parameter in the operator result until the individual is within the bounds and then average it with the current individual. This, however, will bring it closer to the current individual and reduce the explanatory aspect of the operator. A simple rejection method was chosen for this study as this would not decrease the performance of the algorithm and does not affect the Q-table.

Crossover

$$U_{ji}(t) = \begin{cases} V_{ji}(t) & \text{if } rand_{ji} < Cr \text{ or } j = j_{rand} \\ X_{ji}(t) & \text{otherwise} \end{cases} \tag{4-16}$$

The crossover operator is a method to further increase the algorithm's search capabilities. The crossover function that is used is called the binomial cross over function [26]. It chooses a gene from either the original individual or the respective mutated individual at random according to the crossover probability Cr.

The chosen genes are put together, according to equation (4-16), to form the candidate individual $U_i(t)$. For this study the value of Cr is set to 0.5, which means that there is a 50 % chance that half of the original individual will be returned to the candidate. There is however a chance that all the genes originate from the original individual which is not productive for the algorithm. Therefore the probability of a mutated gene being chosen is increased by adding the condition $j = j_{rand}$ where j_{rand} is a random integer within the range of the length of the individual.

Selection

$$\mathbf{Q}_{reward}(R_i(t), F) = \begin{cases} \sum(f_j(\mathbf{X}_i(t)) - f_j(\mathbf{U}_i(t))) & \text{if } \mathbf{U}_i(t) \prec \mathbf{X}_i(t) \quad \forall j \in n_{Obj} \\ -K_{penalty} & \text{if } \mathbf{X}_i(t) \prec \mathbf{U}_i(t) \\ \sum(f_j(\mathbf{X}_i(t)) - f_j(\mathbf{U}_i(t))) & \text{otherwise} \\ & \forall j : f_j(\mathbf{X}_i(t)) > f_j(\mathbf{U}_i(t)) \end{cases} \tag{4-17}$$

The algorithm now must choose between the original individuals and the individuals produced by the algorithm operators. The objective function value of the candidate individual is compared with that of its parent individual to see which individual dominates the other following the rules in section 4-5-2. If the candidate individual dominates the original, the candidate is added to the population while the original is discarded. This is repeated for each pair and afterwards, a new generation is born to be used in the next iteration of the algorithm. Figure 4-7 illustrates the process of the operators used to produce this generation for one individual. The example individuals in this Figure has 5 parameter blocks of which the shade of the parameter block indicates its value.

This means that the algorithm and the selected mutation factor were successful in improving the solution. Therefore a reward, equal to the objective function value improvement, is added to the reward table $\mathbf{Q}_{reward}(t)$ which has the same dimensions as the Q-table. This reward table exists only for this iteration and is used to update the Q-table at the end of the iteration.

If the original individual is dominant, the candidate is discarded and the original individual remains in the population. The algorithm has failed to improve the solution and therefore a pre-defined negative reward $K_{penalty}(t)$ is added to the reward table. In the case that the original individual or candidate is not dominant both individuals are considered valid and the candidate is appended to the population. The reward value added to the reward table is equal to the objective value difference of the improved objectives.

Truncation

At this point the population may contain more individuals than the prescribed population size and must be truncated to reduces its size back to the original population size. This truncation occurs during non-dominated sorting and the crowding distance calculation explained in section 4-5-2.

After the Pareto fronts and crowding distance has been determined for each individual, the Pareto fronts are sorted in ascending order and each Pareto front is added to the new population matrix until a Pareto front does not fit. The individuals are then added according to their crowding distance until the original population number has been reached.

Q-table update

Once all the series of operators have produced the new population, the Q-reward table will be filled with the rewards and penalties given to the mutation coefficient corresponding to the results of the selection operator. This Q-reward table is then combined with the global Q-table in preparation of the next iteration of the algorithm.

A simple update method would be to directly add the reward table per element to the current Q-table. This would work for the first few iterations but due to the varying change in search conditions during the duration of the optimization, mutation coefficients that initially performed well might not produce the same results in later iterations. This is why a learning rate α is introduced to give current rewards priority rather than the rewards achieved in the past.

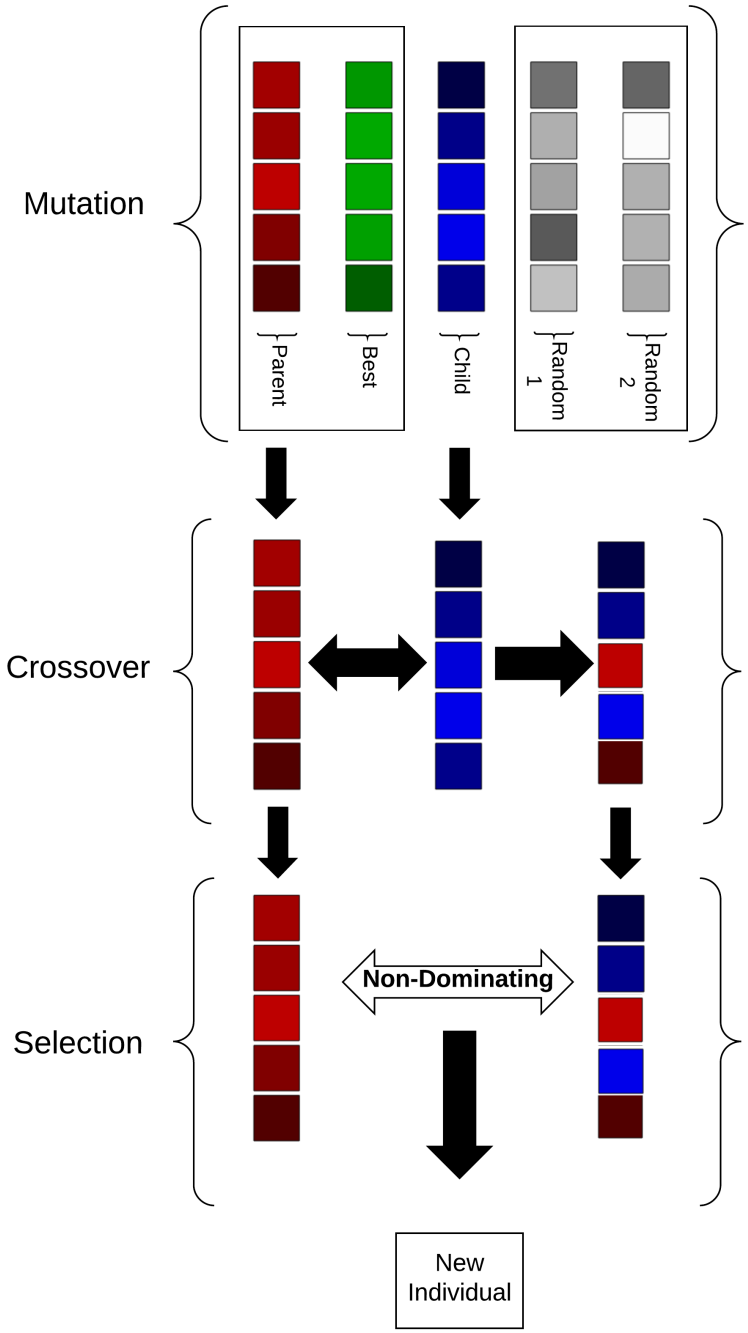


Figure 4-7: An basic illustration of the Mutation, Crossover and Selection operators using example parameter sets of 5 parameters

$$\mathbf{Q}(R_i(t), j) = (1 - \alpha)\mathbf{Q}(R_i(t), j) + \alpha(\mathbf{Q}_{reward}(R_i(t), F) + \gamma \max \mathbf{Q}(R_i(t + 1), j)) \quad (4\text{-}18)$$

Equation 4-18 shows the Q-table update function which is evaluated for every individual in the population of the current generation. Each individual has a rank $R_i(t)$ which corresponds to a row in the Q-table while the columns of the Q-table j correspond to the range of mutation factors Fj. Every probability value in a row is updated with the condition that the value in the Reward table at the same position is not equal to zero.

At this point in the algorithm's work flow, the population of the next generation is already determined and therefore can be ranked before the mutation operator is executed and the values of the Q-table needed. With these two ranked populations a prediction can be made by looking at the maximum possible performance an individuals offspring can achieve and adds this prediction to the reward given to that rank in the Q-table. This ensures that the Q-table is more optimistic by striving to achieve the best possible results in the next generation [26, 29]. This principle is implemented into the update function with $\gamma \max \mathbf{Q}(R_i(t + 1), j)$ where γ is the discounting factor, which controls the influence future predictions have on the Q-table.

4-6 Termination criteria

For multi-objective optimization the change in objective function can be difficult to determine, as new solutions might lie on a Pareto front and may not indicate an improvement in the overall solution. Measurement methods such as the generational distance GD_m, are introduced [30]. Methods such as these determine the progress of a multi-objective algorithm by focusing on the movement and deformation of the Pareto fronts. This can be compared with the standard single objective termination criteria proposed in section 4-5-1 but instead of taking the progress of the individuals into account, the rate of change of the best Pareto front and average Pareto front can be used. A Pareto front with a wide range is desirable as it gives the best insight into the available solutions between the optimum solutions for either objective. Therefore a criteria can be set to only allow the algorithm to terminate if the average crowding distance of the best Pareto front is above a certain level.

$$d_i = \min \|\mathbf{Y}_i - \mathbf{Y}_j\| \text{ for } j = 1, 2., , , N_{current} \quad (4\text{-}19a)$$

$$GD_m = \frac{1}{N_{previous}} \sqrt{\sum_{i=1}^{N_{previous}} d_i^2} \quad (4\text{-}19b)$$

Equation (4-19) calculates the generational spread of each individual in a Pareto front where d_i in equation (4-19a) is the objective difference between an individual in the previous Pareto front, given by index j, to the nearest individual in the corresponding Pareto front in current generation given by index i.

The generational spread indicates the minimum distance that the algorithm has moved the fronts forward and can, therefore, be used as a measurement of the algorithm's progress.

Termination occurs once the average change of the best Pareto front over a defined number of stall generations decreases below a threshold. While the actual crowding distances will not be taken into account for this study, a minimum number of 4 Pareto points in the final solution is required for the algorithm to terminate.

Deterministic algorithms have singular results and therefore instead of calculating the distance of a Pareto front, the termination criteria measures the change in the distance of the objective function value to zero. This ensures that the main termination criteria for both types of algorithms are similar and therefore comparable. Supplementary criteria such as the step size and optimality criteria are left in place as a back up in case the main criteria does not stop the algorithm. This can occur when a deterministic algorithm starts oscillating around its intended solution.

Globally, a maximum iteration limit is placed as a method to save computation time and stop an algorithm that fails to converge to an acceptable solution. Unless divergence takes place, each algorithm has the ability to eventually converge to a solution with endless time to do so, however as the duration of the algorithm is also a measure of its performance, a large number of iterations are not desirable. This lack of performance is reflected in the duration of the algorithm but also its objective function value as the algorithm has not had the chance to converge fully.

4-7 Algorithm implementation

The algorithms described in this chapter were implemented in Matlab with the objective function described in section 4-2. The following list shows the algorithms that were compared in this study:

- The multi-objective function *gamultiobj* was used as a base-line for a multi-objective genetic algorithm based on the basic principles explained in this chapter. This algorithm will be referred to as Multiple Objective Genetic Algorithm (MOGA).
- The *lsqnonlin* function, which will be referred to as Mixed Least Squares (MLSQ), was used as a base line for a deterministic algorithm and is a non-linear least mean square curve fitting algorithm accepting only one objective which was taken as the average of the normalized objective functions provided by the simulation.
- The goal attainment algorithm was implemented using sequential combination of the *lsqnonlin* function to determine the independent objective goals and the *fgoalattain* function put forth in section 4-4-2.
- The Demo-TDQL algorithm was completely implemented programmatically according to the [26] with modifications with respect to the automotive problem such as bounds and stopping criteria.

Standardisations were implemented to be able to compare the performance of the algorithms with each other. This can be a daunting task considering the differences in algorithm principles and datasets. Where common parameters exist, these were made equal.

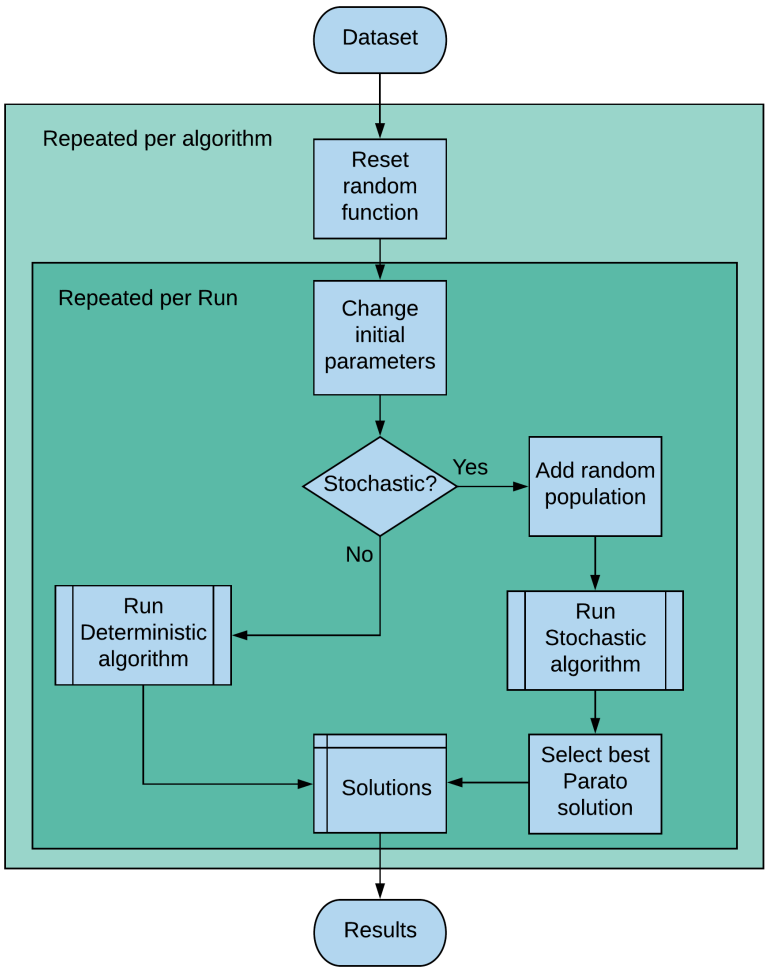


Figure 4-8: Flow chart of the algorithm comparison run procedure

The population size for the MOGA and the DEMO algorithms were kept constant at 20. The size was difficult to define as a smaller population size would work in favour of the adaptive component featured in DEMO as the number of generations would increase leading to more information for the Q-table. However, a small population would hinder the searching ability of MOGA. The finite difference step which is used by the deterministic algorithms MLSQ and Goal attain was chosen to be 1e-3.

The termination criteria, which is the biggest factor of the algorithm's final accuracy, was chosen as the change in objective function and/or Pareto front per iteration. This act as the main termination criteria for all algorithms and was standardized to be a change less than 1e-3. A stricter criterion was investigated but did not show a significant increase in accuracy but did increase the duration of the algorithms. Furthermore, a final limit placed on all algorithms is the number of function evaluations allowed per algorithm run. This was set at 800 iterations and would stop if this were reached. When this occurs, it is considered an outlier result and is important as this is an indication of the algorithms reliability.

The initial parameter set for deterministic algorithms is standardized for all algorithms during a repetition of a dataset. This means that it is also added to the initial population of a stochastic algorithm to ensures that if the chosen initial parameter set were already the optimal solution, both algorithm types will have access to it and converge just as quickly. Deterministic algorithms produce the same results when the settings and inputs are kept constant and as the initial parameter set is pre-determined and is usually an educated guess. To simulate the uncertainty of a guess based on previous data knowledge, a random element was added to the initial parameter set per algorithm repetition. This simulates human uncertainty with up to 20%.

To further standardize the inputs for comparison, the random seed function in Matlab set to produce the same sequence of values. This ensures that the algorithms receive the same random numbers and the behaviour of the algorithms can, therefore, better be compared.

Chapter 5

Results

This chapter will examine the results of the execution of the four algorithms implemented in the previous chapter. The algorithms were run sequentially on each of the datasets acquired during experimental testing which compromise of 6 repetitions for the step manoeuvre and 4 repetitions for the steady-state manoeuvre. Multiple runs were executed for each dataset partly due to the nature of the stochastic algorithm,Differential evolution for Multi-objective optimization (DEMO) and Multiple Objective Genetic Algorithm (MOGA), but also as a way to vary the initial dataset of the deterministic algorithms, Goal attain and Mixed Least Squares (MLSQ). Many aspects of the algorithms were standardized using the same type of termination criteria and have equal algorithm parameters where this is applicable. This is done to ensure that the results are comparable. Focus will be placed on the objective function of the algorithms, the Normalized Root Mean Squared Deviation (NRMSD). The computational duration of the algorithm as well as the reliability of the resulting parameter set will also be evaluated.

Validation of the resulting parameter sets is done by means of cross-validation between the experimental datasets to determine which type of test manoeuvre produces an accurate and flexible solution. A dedicated validation dataset mentioned in section 3-3-3 is then used to evaluate the dataset further as this dataset better reflects normal operation. This chapter is concluded with an overall discussion of the results including the limitations of the datasets and algorithms observed.

5-1 Preliminary results

Before the primary results are evaluated, preliminary investigations were done to get a better understanding of aspects of the implemented algorithms and how it interacts with the parameter search space. Firstly the influence of the initial position on the outcome of deterministic parameters will be investigated. This is followed by a look into two different methods of using the available data by deterministic algorithms. This is related to the choice of algorithm to determine the goals of the goal attain algorithm. Lastly the sensitivity of the mutation operator on genetic algorithm results will be investigated. This is important to understand the need of the adaptive component implemented in the DEMO algorithm.

5-1-1 Deterministic initial position

The biggest disadvantage of deterministic algorithms mentioned in this study and in the literature is the dependency on a pre-defined initial parameter set. Deterministic algorithms tend to prematurely converge into local minima and therefore produce inaccurate solutions. To investigate if this case applies for this set of data, a simple test was done using the deterministic *lsqnonlin* algorithm.

Starting with the initial parameter set values chosen for this thesis (table 4-1), each parameter is either increased or decreased by 10 % of its bounded range. After this change the parameter set is used as the initial parameter set for the *lsqnonlin* algorithm. The change in resulting parameter solutions is recorded before moving to the next parameter in the set. This is after resetting the previous parameter to its original value.

Table 5-1: The change in the solution parameter set after the initial position has been varied by 10% per parameter

	Percent change in resulting parameter set	
	$\mathbf{X}_0 - 10\%$	$\mathbf{X}_0 + 10\%$
RLF	2.3%	2.1%
RLR	7.6%	4.3%
DR	25.3%	8.0%
CR	3.0%	9.4%
BR	2.7%	7.5%
ER	6.8%	3.5%
SvR	4.1%	5.1%
ShR	7.0%	4.5%
DF	8.2%	18.8%
CF	1.7%	1.0%
BF	1.8%	12.0%
EF	42.1%	2.2%
SvF	6.7%	4.5%
ShF	4.9%	5.5%

Table 5-1 shows the percentage of the euclidean distance changed by the parameter set relative to the initial parameter set vector $\mathbf{X}_0$. The results show that there are other areas of local

minima which influence the path of a deterministic algorithm. If there was only one possible solution in the area, the resulting parameter set produced by the algorithm would remain the same and should not change with a different initial parameter set. The presence of these local minima may stem from the inaccuracies in the measurement data and/or due to the inabilities of a simplified model.

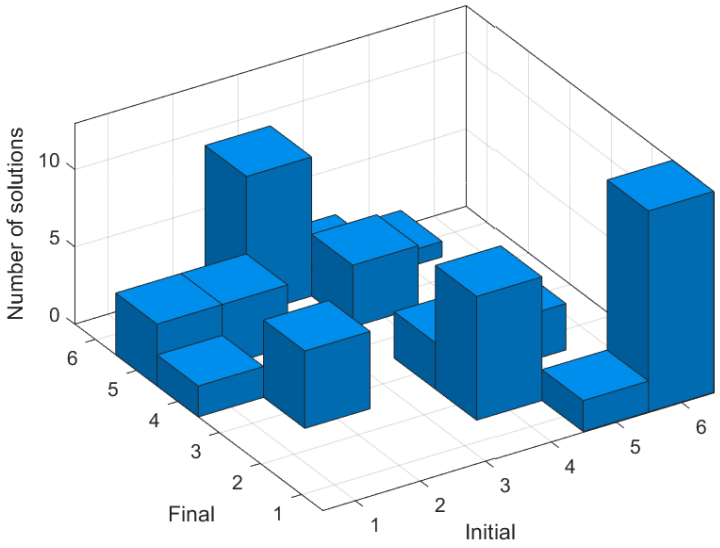


Figure 5-1: Histogram illustrating the areas of local minimal that attract solutions starting from different initial parameter sets

To prove the presence of local minima in the area, the results of the main algorithm iterations covered later in this chapter were used. The initial and final positions of the parameter set produced by the goal function of the goal attain algorithm were analysed to see if a pattern could be identified. The initial and final parameter sets were grouped into clusters with similar values surrounding a centroid parameter set which is the component-wise median of the cluster. Figure 5-1 shows a histogram plot of the clusters to which the initial and final parameter sets belong. These clusters are ordered in the manner of distance from each other and show how different starting points produce different results. This reaffirms the disadvantage of a deterministic algorithm when it comes to this particular measurement data and model optimization.

5-1-2 Lsq and fmincon

The goal attainment algorithm, *fgoalattain*, requires optimum values for both the lateral acceleration and yaw-rate objective to act as goals optimize towards. There are two single objective algorithms available in *Matlab* that can be used for this purpose. The first is the *lsqnonlin* algorithm which is based on the least squares method and uses the raw data from the simulation and the experiments to fit the curve. The other is the *fmincon* algorithm which uses the value calculated by the objective function instead of the raw simulation outputs. The

advantage of *fmincon* is that it directly uses NRMSD value and does not have to be calculated afterwards such as with *lsqnonlin*.

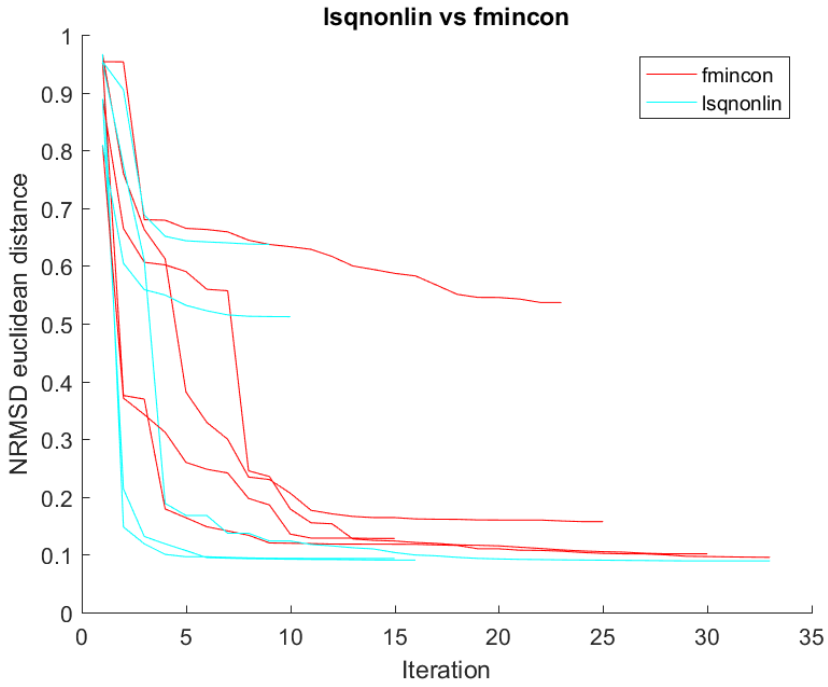


Figure 5-2: Objective function value progress across iterations for the fmincon and lsqnonlin Matlab functions

Figure 5-2 shows the change in NRMSD per iteration repeated at 5 randomly varied parameter sets surrounding the initial parameter set. The least squares objective function performs better at achieving the objective but has a higher chance of not converging to valid solution. This instability is found to be the result of the algorithm diverging towards the parameter bounds. Once situated at the bounds, the optimization converges to an unusually high local minimum.

This is a problem if the solution was only based on this result. However, as these solutions will be used as goals for the final *fgoalattain* algorithm, these erroneous goals will be quickly overachieved and disregarded. The *fgoalattain* is expected to be require a longer duration to optimize the problem without goals in place to determine which objective takes priority. The *lsqnonlin* is used for this study as the advantages of accurate goals outweigh the advantages of averagely less accurate goals but with a higher reliability.

5-1-3 Mutation operator coefficient sensitivity

One of the main parts of this study was to reduce the number of critical algorithm parameters that the user needs to choose before an algorithm is run. The adaptive algorithm is a method to reach that goal by providing a way to determine parameters during the optimization. Figure 5-3 shows the average objective distance of the DEMO algorithm per generation. The mutation coefficient, F_j, is varied and different characteristics are observed. For a low

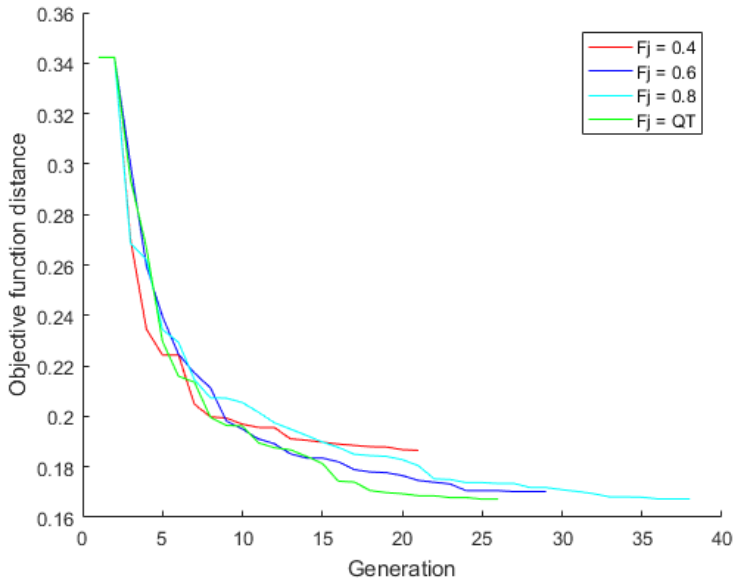


Figure 5-3: Solution progression with respect to mutation coefficient F_j

mutation coefficient, the algorithm converges quickly but stalls prematurely, while for a high mutation coefficient, the algorithm does produce better results but after a longer period of time. The figure also illustrates how the Q-table modification of the genetic algorithm, which varies the mutation coefficient, improves these results by exhibiting both characteristics.

It is also noted that the number of generations needed to achieve the same objective function value is reduced up to 15 generations when adapting the mutation coefficient in real time. This does not provide much improvement for the current small scale problem that takes on average 3 s per generation. However, a more complex version of the vehicle and/or tire model could profit from this decrease in required computational power to provide a solution faster that is just as accurate.

5-2 Algorithm results

As explained in section 4-7 all algorithms were run consecutively using the standardized settings and inputs laid out in previous sections. The results of these comparative runs will be presented and analysed in this section. Focus will be placed on the objective function values, duration, and the quality of the solution produced by these algorithms.

5-2-1 Objective function value

The objective of the optimization algorithms is to reduce the NRMSD of both the model outputs by iteratively varying the model parameters. A optimization algorithm that can reliably produce an accurate solution is the biggest indicator of its performance.

Initially all algorithms were repeated 12 times on all datasets. As will become clear later in this section, the step manoeuvre was further repeated to increase the significance of the

statistical tests. The results of which will determine if one algorithm performs exceptionally better.

Table 5-2 contains the mean and standard deviation of the respective NRMSD euclidean distances values which will be regarded as the objective value unless otherwise stated. For the algorithms that produce a set of Pareto solutions, the solution with the least NRMSD euclidean distance is chosen as the objective value for that run.

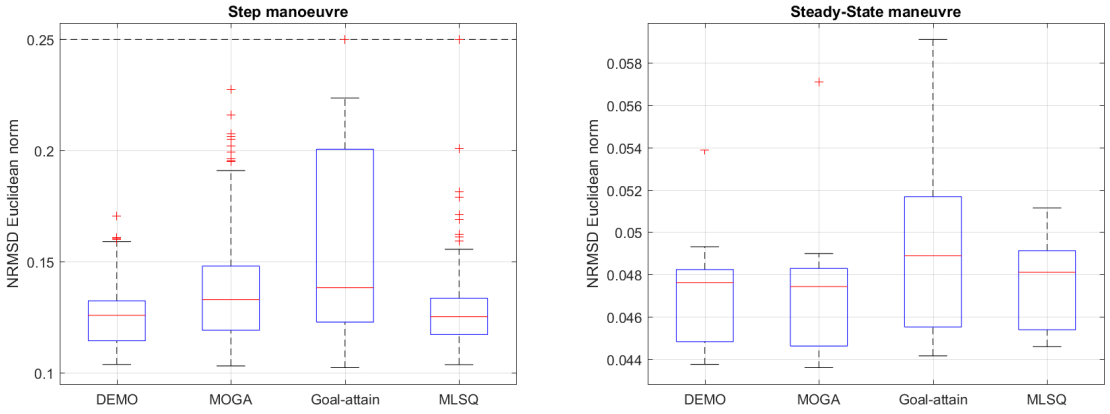
Table 5-2: The NRMSD euclidean distance values for the datasets per algorithm with standard deviation in brackets

Test	Direction	Set	DEMO	MOGA	Goal attain	MLSQ
Step (N=80)	**Left**	1	0.1372 (0.0115)	0.1573 (0.0212)	0.1672 (0.0528)	0.1417 (0.0172)
		2	0.1284 (0.0065)	0.1393 (0.0149)	0.1946 (0.2233)	0.1310 (0.0086)
		3	0.1113 (0.0056)	0.1147 (0.0081)	0.1721 (0.1706)	0.1190 (0.0072)
	Right	1	0.1327 (0.0095)	0.1488 (0.0188)	0.5118 (0.5254)	0.2214 (0.2325)
		2	0.1118 (0.0054)	0.1174 (0.0089)	0.1926 (0.2523)	0.1095 (0.0038)
		3	0.1291 (0.0052)	0.1375 (0.0099)	0.4967 (0.3068)	0.2147 (0.2245)
Steady-state (N=12)	**Left**	1	0.0487 (0.0017)	0.0489 (0.0026)	0.0522 (0.0034)	0.0492 (0.0008)
		2	0.0449 (0.0010)	0.0449 (0.0011)	0.0465 (0.0024)	0.0455 (0.0010)
	Right	1	0.0616 (0.0087)	0.0915 (0.0209)	0.0513 (0.0013)	0.0516 (0.0017)
		2	0.0574 (0.0087)	0.0799 (0.0177)	0.0458 (0.0027)	0.0473 (0.0013)

Figure 5-4 shows two box-plots illustrating the results for the step and steady-state manoeuvre of all the respective datasets combined. This representation of the objective distance data shows that the DEMO and MLSQ algorithm produce the better results on average for the step manoeuvre. The results for the steady-state test are not so clear but clearly indicate that the goal attain algorithm does not perform as well as the other algorithms.

Manoeuvre verification

It was noted that the objective values for the steady-state manoeuvres are significantly lower than that of the step manoeuvre. This was most likely due to the fact that the model is better equipped to model steady-state manoeuvres and therefore provides a solution with a better fit. In light of the added complexity of modelling a dynamic step manoeuvre, this makes it difficult for the model to simulate everything accurately and therefore a larger NRMSD is expected on average.

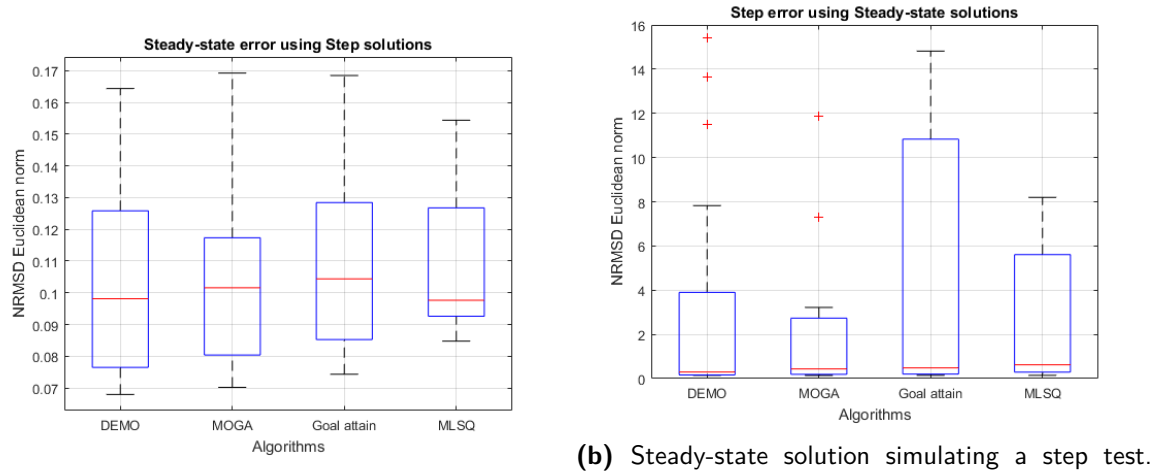


(a) Step manoeuvre N = 80

(b) Steady-state manoeuvre N = 12

Figure 5-4: Box-plot of the Objective function distance values per manoeuvre and algorithm

In light of this fact, an investigation is done to determine how flexible the resulting parameter sets are relative to the manoeuvre used to obtain them. Figure 5-5 shows a plot of the cross validation for both steady-state and step manoeuvres. The parameter sets produced by the algorithm solutions that were retrieved using the step manoeuvre are used to simulate a steady-state manoeuvre and vice versa.



(a) Step solutions simulating a steady-state test

(b) Steady-state solution simulating a step test. Note the y-axis shows a drastically larger scale than normal

Figure 5-5: Cross validation of algorithm results

The results from these cross validation simulations show that the parameter set solutions from the steady-state optimization produce large errors when simulated on a more dynamic dataset. While some parameters produces comparable results to the step manoeuvre solutions, a large part of the simulations produce large errors which lowers the reliability of the manoeuvre as an optimization dataset.

Table 5-3 shows the objective function of the best simulations for either cross test. It shows

Table 5-3: The minimum Objective distance values of the cross-validation for each algorithm compared with the average of the original results retrieved in the previous section.

Cross validation	DEMO	Goal attain	MLSQ	MOGA	Original average
Step	0.055	0.059	0.055	0.061	0.05
Steady-state	0.24	0.263	0.285	0.18	0.15

how the best solutions for either cross test fall short of the average objective function value produced by the original optimization. This was to be expected as model optimization optimizes the parameters for a specific dataset producing a situation where the model is over-fitted to the particular dataset. This produces large errors when this model is used to simulate other datasets that don't resemble the original dataset.

Statistical analysis

Besides the conclusions that are made using the visualized data, statistical tests provide a better indication on how the algorithms perform with respect to each other. For this reason, tests were carried out to determine if any algorithm performed significantly better and a pair-wise investigation for a better indication of which algorithms these are [31].

To improve the significance of the hypothesis tests, the number of algorithm runs per dataset for the step manoeuvre was increased to 80. This was only done with the step manoeuvre, as the results of the steady-state test have shown that optimization solutions do not show reliable flexibility to simulate different vehicle manoeuvres.

The Friedman's test is a two-way non-parametric test which investigates problems with two different variables, in this case the dataset runs and the different optimization algorithms. This test is frequently used as it adjusts for differences in algorithm runs but focuses on the objective value difference between the algorithms. The null hypothesis states that all the medians for the different algorithms are equal and is rejected when the probability of which drops below the 5% threshold [32].

Table 5-4: The rank values resulting from the Friedman test on the resulting objective values. Ranks indicate the performance of the algorithm with a lower rank indicating a better rank. The null hypothesis probability indicates the probability of the null hypothesis being true.

	Lateral acceleration	Yaw rate	Objective value
Null hypothesis probability	2.82e-49	1.53e-33	1.00e-40
Critical Difference	1.29	1.29	1.29
DEMO	1.99	1.98	2.00
MOGA	2.86	2.86	2.89
Goal attain	3.01	2.83	2.91
MLSQ	2.13	2.34	2.21

Table 5-4 show the results of the Friedman test where the values per algorithm are its mean rank. If an algorithm produces a result with a mean rank lower than its counterparts by the critical difference, then the null hypothesis, is rejected. DEMO and MLSQ algorithm rank lower than the other two, but no one of the algorithms performed decisively better. The

null hypothesis does have a very low probability overall which rejects the hypothesis that the median values are equal.

This result leads to a more specific investigation of the algorithms. This is done by applying the Wilcoxon rank sum test which is a robust test to compare two independent populations. It is robust as it is able to investigate skewed distributions and make a normal approximation of the data [32]. The test is run as a single tail test with a null hypothesis stating that algorithm **X** does not perform better than algorithm **Y**.

Table 5-5: Single tailed Wilcoxon rank sum test. The algorithms were tested pair-wise using the objective values. The results in this table show the probability of the null-hypothesis

		Y			
		DEMO	MOGA	Goal attain	MLSQ
X	DEMO	0.5	**9.8e-17**	**2e-22**	0.14
	MOGA	1	0.5	**6.6e-06**	1
	Goal attain	1	1	0.5	1
	MLSQ	0.86	**1.8e-10**	**1.6e-16**	0.5

Table 5-5 shows the results of this test where the results in bold indicate significant results. It confirms that the DEMO and MLSQ perform better than the MOGA and Goal Attain algorithms. This also shows the the Goal attain algorithm performs the worst out of the group.

5-2-2 Computation duration

Another measure of algorithm performance is the duration that it took for an algorithm to converge to a solution. The duration is used as it gives a practical indication of the algorithms computational efficiency.

Figure 5-7 is histogram illustrating the distribution of the objective and duration results per algorithm. Both DEMO and MOGA have runs concentrated at the lower left side of the distribution indicating a good objective value and short durations to produce a solution.

The Goal-attain algorithms perform visibly worse, having a larger deviation from the optimal region and shows a certain amount of instability with many outliers. The algorithm was already expected to have a longer computational time due to the fact that it first needs to determine the objective goals before the *fgoalattain* algorithm can be executed to produce a global solution. The MLSQ algorithm does not have this problem is indeed shows a better performance and consistently produces low objective values but with longer durations than that of the stochastic algorithms. The MLSQ algorithm does show singular outliers when it comes to the objective value.

Just as with the objective values, the duration was subjected to the Wilcoxon test. The results show that when it comes to the duration of the optimization, the MOGA performs better than the DEMO algorithm.

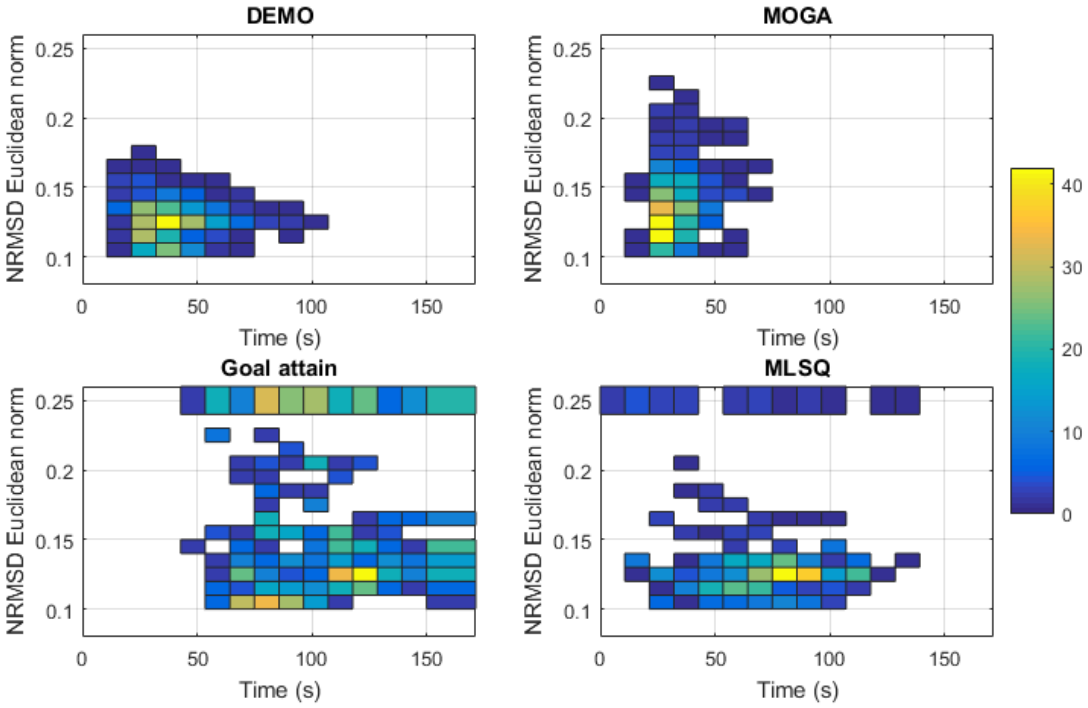


Figure 5-6

Figure 5-7: A histogram showing the algorithm results sorted into objective value and time where an optimal result should have a low NRMSD within a short duration.

Table 5-6: Single tailed Wilcoxon rank sum test run pair-wise on the duration. The results in this table show the probability of the null-hypothesis

		Y			
		DEMO	MOGA	Goal attain	MLSQ
X	DEMO	0.5	1	**2e-151**	**3.9e-82**
	MOGA	**5.6e-14**	0.5	**2.6e-158**	**1.1e-107**
	Goal attain	1	1	0.5	1
	MLSQ	1	1	**6.1e-64**	0.5

5-2-3 Pareto results

While the previous results are based on the lowest magnitude of the objective function, the entire Pareto range must be taken into account. Figure 5-8 shows the solutions of the step manoeuvre for one dataset. It is immediately clear that the DEMO algorithm produces the widest Pareto front compared to the other stochastic algorithm MOGA. The Pareto front shows the range of solutions that are considered just as optimal for either the lateral acceleration or the yaw-rate. The difference in lateral acceleration accuracy between the Pareto front extremes can be up to 20 %, which shows that while an overall accuracy can be achieved, the position of the solution on the Pareto front can make a difference in model behaviour during a simulation.

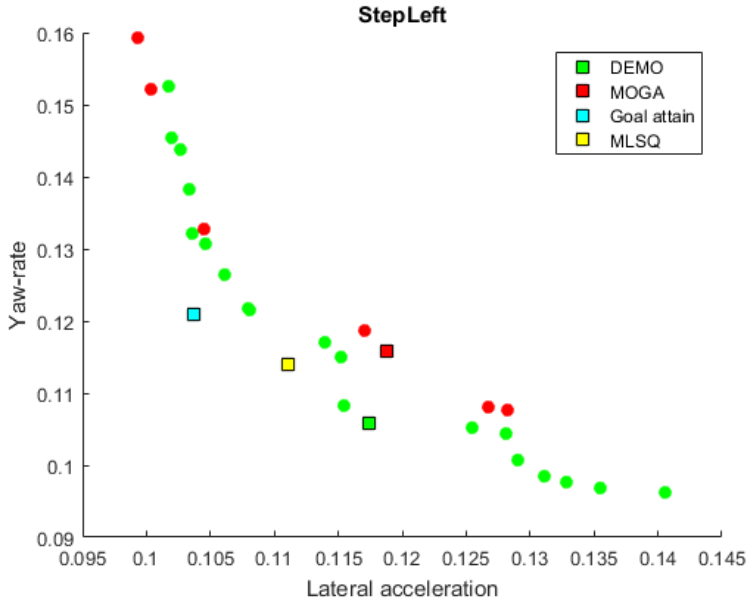


Figure 5-8: An example of the Pareto fronts based on one of the Step manoeuvre algorithm results. The solutions resulting from the deterministic algorithms are also included for comparison

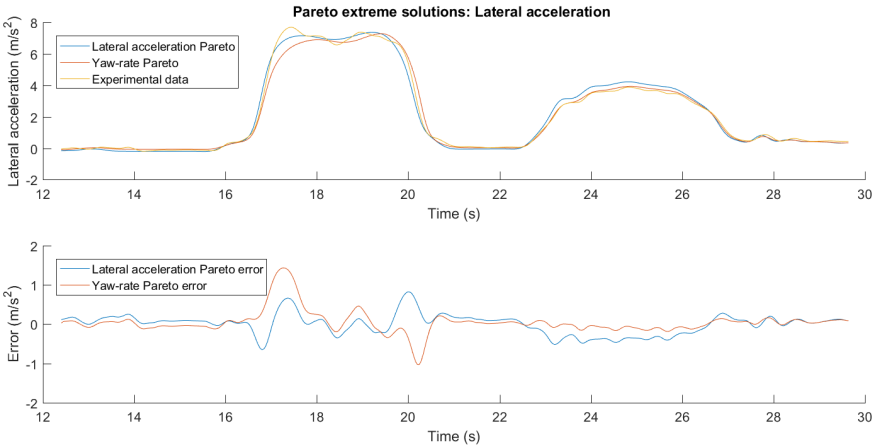


Figure 5-9: Lateral acceleration Pareto extreme

Figures 5-9 and 5-10 show how the simulation results differ when Pareto solutions from either extreme is used by the model. This shows that while not very large, the difference between the two solutions is capable of producing an error in either objective output data.

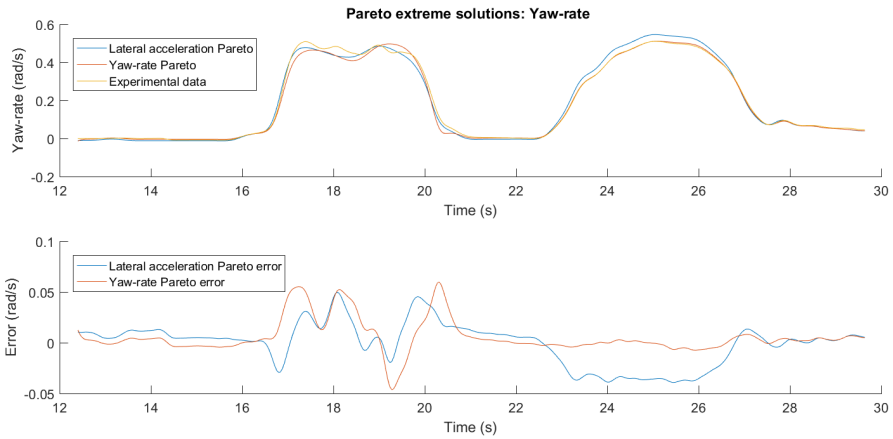


Figure 5-10: Yaw-rate Pareto extreme

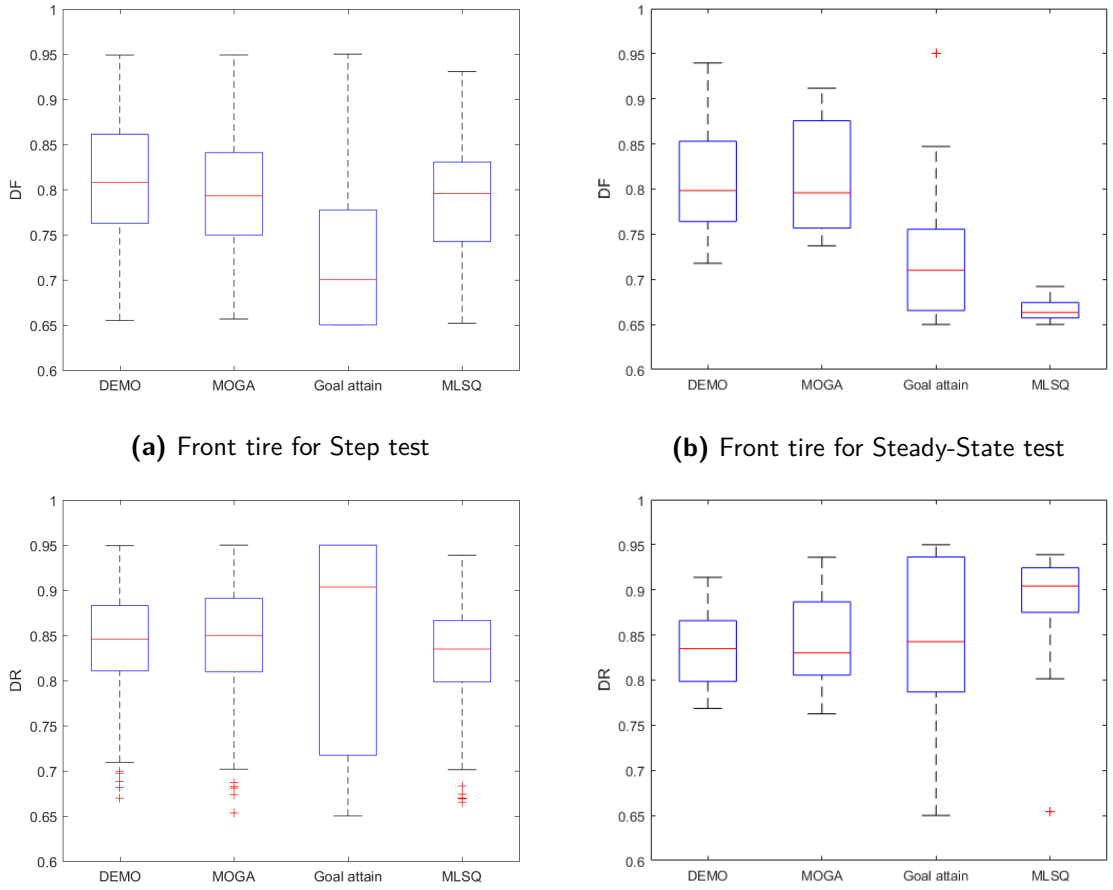
5-2-4 Parameter solution

After having determined the optimal solution per algorithm, the parameters of these solutions can be evaluated to see how these vary. The ultimate optimal solution is unknown and therefore a general investigation must be made to see if an optimal parameter set has been frequently determined by the algorithms. The parameter set consists of 7 parameters for the front and rear wheels which make a total of 14 parameters. As it is difficult to illustrate the positions of the solutions in relation with each other within the search space, individual parameters were simulated by the vehicle model to produce the results seen in figure 5-11.

The Magic formula coefficient D is a physical principle as it describes the friction between the tire and the road. This coefficient depends on the road and weather conditions and therefore can not be predetermined with respect to a specific tire. Ideally the algorithms should converge to the same friction coefficient as this physical principle should be the same across all datasets. Considering that the experimental tests were done under wet conditions, a lower friction coefficient is expected and therefore a 0.75 value was used in the estimated initial parameter set.

Figure 5-11 shows that the friction coefficient range per algorithm for both tests and tires. The algorithms estimate a higher friction coefficient for the rear tires and a relatively low value for the front tire. This could be due to the overall handling characteristics of a front wheel driven vehicle that exhibits understear, having less grip at the front tires. For the step manoeuvre, the averages lie around the same value range while the deviations of the data vary greatly. This could be due to the nature of a dynamic test which tend to experience a large range of friction values during the rapid changes of state experienced by the vehicle. The steady-state parameters are more precise as part of the tests design is to determine the tire friction which varies less as the slip values rise slowly.

Both DEMO and MOGA do not trend to a specific value for coefficient D, while the goal attain algorithm performs the worst by producing a large range of values. MLSQ algorithm shows significantly lower variance which may point to a higher parameter accuracy although outliers do occur. The steady-state manoeuvre was performed on a wetter roads which coincides with the low averages produced by the algorithms.



(a) Front tire for Step test

(b) Front tire for Steady-State test

(c) Rear tire for Step test

(d) Rear tire for Steady-state test

Figure 5-11: Friction coefficient D results

Figures 5-12 and 5-13 show the simulation outputs for the lateral acceleration and yaw-rate using the best results produced by the optimization algorithms. The lower plot, in both figures, show the error with respect to the y axis. This illustrates the differences of the algorithms in a clear manner.

As a verification of the model with respect to the test data, the parameter solutions were used to calculate the linear cornering stiffness of each tire and subsequently the understeer characteristics of the vehicle model as explain in chapter 3. Figure 5-14 shows the understeer characteristics of the best solutions for each algorithm for all datasets. These are compared with the experimental data and a loosely fit curve to indicate the trend in the data.

All algorithms show the same understeer characteristic while most lie below the data trend. This can be explained due to the previously mentioned non-linear characteristics which are not taken into account by the understeer characteristics. Only the MOGA algorithm shows a steeper curve which may be due to apparent over estimation of coefficient D therefore exhibiting less of an understeer characteristic.

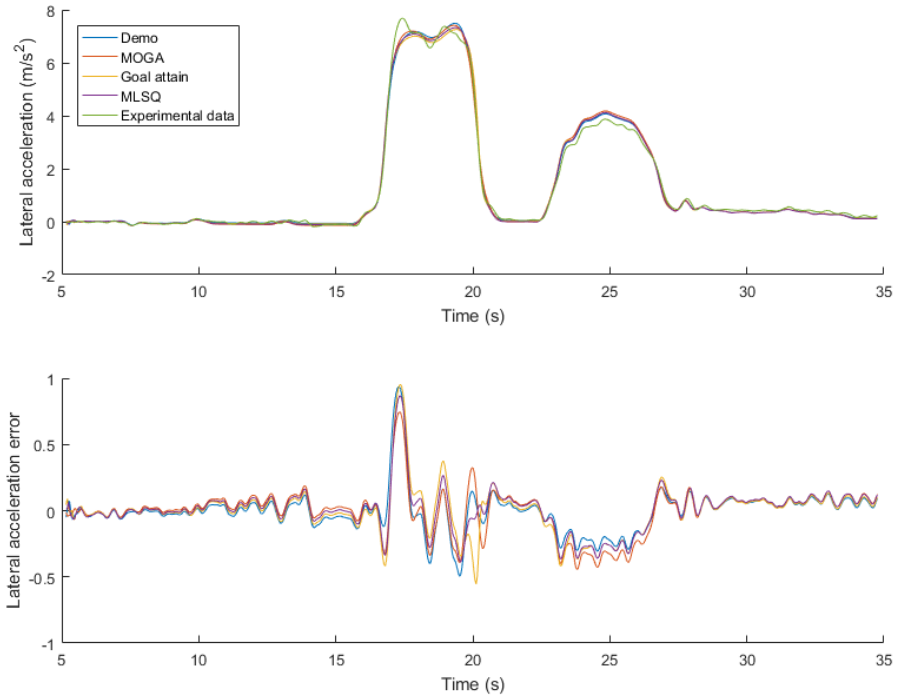


Figure 5-12: Lateral acceleration plot

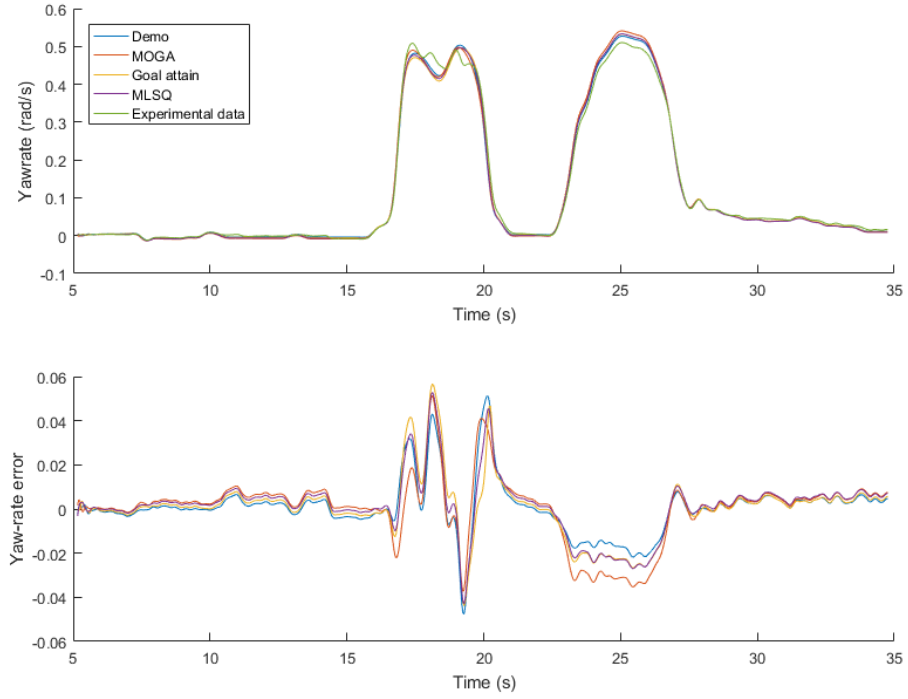


Figure 5-13: Yaw rate plot

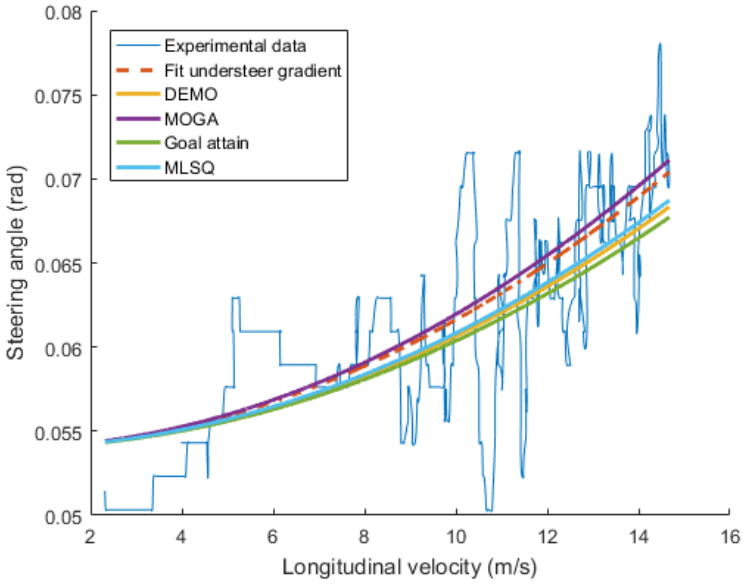


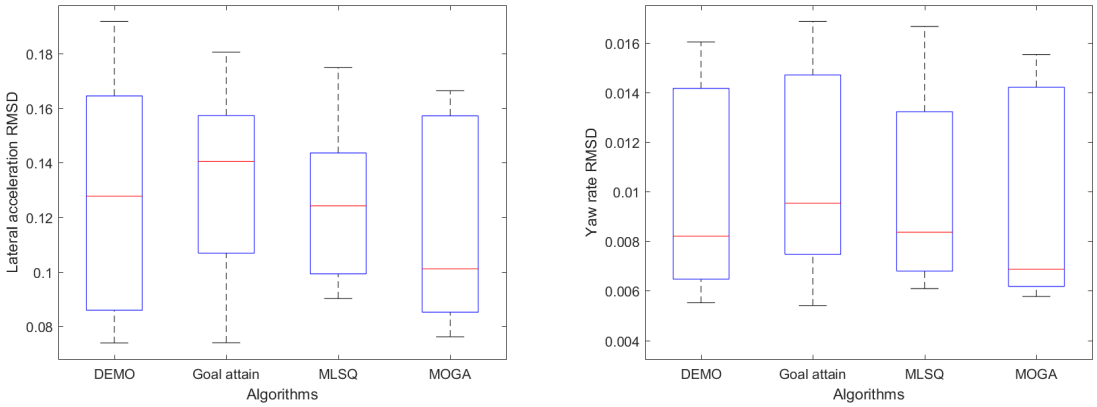
Figure 5-14: Understear gradient of Solutions

5-3 Validation

Validation tests the models to investigate if they are able to properly simulate the vehicle manoeuvres. If this is the case, the optimized models produced by the algorithms are reliable enough to be used as simulation tools. While the ideal method of validation would be to have tire data retrieved in a laboratory setting, this data is unavailable. Validation is done using other datasets that were not used in the optimization process.

As mentioned in chapter 3 the return journey from the test site back to the university was recorded. This dataset, reflecting general driving conditions, acts as a method of validation of the parameters set produced by the four algorithms. This data is independent of the dataset used to optimize the model parameters and therefore a good measure of the results. Having concluded in the previous section, that the solutions produced by the step response data show a great flexibility in simulating other datasets, these solutions have primarily been used for this validation.

Figure 5-15 shows the box-plots for the lateral acceleration and the yaw-rate simulated over all intervals of the validation dataset. Interestingly, the DEMO algorithm does not produce consistent accurate results which is alternatively done by MOGA. The other algorithms do not show any decisively different results although the goal attain algorithm again does not perform well. The objective function used in the previous analysis is not used on these results due to the large differences in the measurement mean which is used in the normalization of NRMSD. Therefore the values in the plot and in Table 5-7 are given in Root Mean Squared Deviation (RMSD) for each lateral acceleration and yaw-rate. The table shows the individual results for each validation measurement interval, which are parts of the dataset recorded on a return journey from the test site as explained in section 3-3-3. The 5th internal was recorded while driving on a highway at high speed and is the reason for the high maximum values in box-plot 5-15.



(a) Lateral acceleration objective plot per algorithm **(b)** Yaw rate objective plot per algorithm

Figure 5-15: Objective value plots for the Validation datasets

Table 5-7: RMS of the validation test set simulated with results produced by step response test set

		DEMO		**Goal-attain**		**MLSQ**		**MOGA**	
Dataset	**Interval**	LA	YR	LA	YR	LA	YR	LA	YR
Left 1	1	0.1296	0.0119	0.1316	0.0119	0.1435	0.0131	0.1257	0.0121
	2	0.0739	0.0055	0.0740	0.0054	0.0935	0.0067	0.0762	0.0058
	3	0.0858	0.0059	0.0863	0.0059	0.1058	0.0068	0.0877	0.0062
	4	0.0571	0.0032	0.0566	0.0032	0.0788	0.0044	0.0611	0.0036
	5	0.2755	0.0120	0.2719	0.0118	0.2925	0.0127	0.2803	0.0122
Left 2	1	0.1528	0.0147	0.1573	0.0150	0.1499	0.0146	0.1437	0.0142
	2	0.0859	0.0065	0.0930	0.0068	0.0902	0.0068	0.0892	0.0068
	3	0.0975	0.0065	0.1068	0.0068	0.1033	0.0068	0.1031	0.0068
	4	0.0599	0.0034	0.0683	0.0038	0.0666	0.0039	0.0664	0.0039
	5	0.2403	0.0110	0.2518	0.0113	0.2534	0.0114	0.2558	0.0115
Left 3	1	0.1501	0.0142	0.1555	0.0147	0.1678	0.0164	0.1519	0.0146
	2	0.0788	0.0059	0.1048	0.0075	0.1206	0.0087	0.0789	0.0061
	3	0.0851	0.0056	0.1195	0.0075	0.1419	0.0090	0.0851	0.0058
	4	0.0531	0.0029	0.0860	0.0049	0.0998	0.0058	0.0513	0.0029
	5	0.2257	0.0103	0.2841	0.0124	0.2951	0.0130	0.2136	0.0100
Right 1	1	0.1665	0.0159	0.1498	0.0148	0.1396	0.0132	0.1633	0.0155
	2	0.1917	0.0123	0.1592	0.0105	0.1277	0.0084	0.1648	0.0107
	3	0.1909	0.0115	0.1591	0.0099	0.1323	0.0083	0.1613	0.0098
	4	0.1924	0.0112	0.1570	0.0093	0.1205	0.0071	0.1638	0.0095
	5	0.3195	0.0132	0.2883	0.0122	0.2763	0.0118	0.2771	0.0117
Right 2	1	0.1644	0.0160	0.1744	0.0166	0.1748	0.0167	0.1571	0.0149
	2	0.1001	0.0072	0.1231	0.0084	0.1016	0.0073	0.0852	0.0063
	3	0.0942	0.0066	0.1138	0.0075	0.0992	0.0068	0.0852	0.0059
	4	0.0905	0.0052	0.1218	0.0069	0.0890	0.0050	0.0650	0.0037
	5	0.1661	0.0084	0.1813	0.0089	0.1672	0.0085	0.1896	0.0091
Right 3	1	0.1723	0.0160	0.1804	0.0169	0.1621	0.0151	0.1663	0.0155
	2	0.1302	0.0086	0.1534	0.0100	0.0904	0.0065	0.0991	0.0070
	3	0.1258	0.0078	0.1493	0.0091	0.0927	0.0061	0.0989	0.0064
	4	0.1227	0.0069	0.1497	0.0085	0.0693	0.0038	0.0819	0.0046
	5	0.2227	0.0100	0.2487	0.0109	0.2123	0.0099	0.2041	0.0096

5-4 Discussion

This section will summarize and discuss the specific results in relation with each other.

- The presence of local minima throughout the search space can be observed in the results of the friction coefficient parameter DR and DR in section 5-2-4. All algorithm produced solutions with a wide range of parameter values indicating many areas of local minima which reinforces the results in section 5-1-1.
- The deterministic algorithms encounter problems when diverging towards the bounds of the search space. This occurrence was witnessed in section 5-1-2 where both algorithms stagnated at a high objective function before terminating. This irregularity may skew the results of the algorithm's performance, as seen in the standard deviations of the goal attain algorithm in table 5-2. If it wasn't for these outliers, the chances would have increased that the MLSQ algorithm would statistically perform at par with the DEMO algorithm.
- The parameter sets produced by while using the steady-state manoeuvre did not perform well simulating the step manoeuvre. This due to over-fitting and the lack of dynamic excitation to properly optimize the parameters for other handling situations. This over-fitting can also be observed in section 5-2-4 where the friction coefficients are more precise than that of the step manoeuvre.
- The DEMO algorithm was on average the better performing algorithm when the optimization objectives are concerned. Its performance and reliability are matched by the MLSQ algorithm which it failed to reject the null hypothesis of the single tailed Wilcoxon test. The DEMO did, however, reject the null hypothesis of the computational duration comparison.
- The MOGA algorithm did not stand out when its objective results were compared, having only outperformed the goal attain algorithm. However, it did prove to converge to a solution significantly quicker than the other algorithms which were unexpected as the DEMO algorithm has an adaptive component which should help it to converge quicker. This may be due to the small scale of the model being optimized and that the Temporal Difference Q Learning (TDQL) component needs more iterations to properly build a Q-table to improve its speed. In general for similar objective function values, the DEMO and the MOGA algorithm performed much better than the deterministic algorithms and have a more consistent objective value vs time distribution.
- The stochastic multi-objective algorithms produce a solution with multiple parameter sets called a Pareto front. This gives them an advantage as more information is gathered about the search space and the positions of different solutions. Comparing two Pareto solutions with the maximum difference in objective values may produce a lateral acceleration error of up to $0.5\,\mathrm{m/s^2}$ and a yaw-rate error up to $0.04\,\mathrm{rad\,s^{-1}}$5-2-3. This may not seem much but collectively it can add up when simulating large datasets.
- The simulation of the validation dataset using the parameter solutions test the accuracy and flexibility of the resulting vehicle model. The results of which were observed in the

last section of this chapter. It showed that the MOGA solutions produced the overall best results having the lowest median value among all the other algorithms. This does contradict the results of the previous analysis of the results. A reason for this may be that the MOGA algorithm performs averagely and thus is not prone to over-fitting to a dataset. This allows the solutions to be able to be applied to different datasets as it stops short of optimizing a model perfect for a particular dataset.

Chapter 6

Conclusion and Recommendations

6-1 Summary

The research question for this thesis is to determine if an adaptive genetic algorithm performs better estimating vehicle lateral dynamic model parameters compared to deterministic algorithm using user defined initial parameters. To answer this research question this study covered all the steps required to produce an accurate model to simulate lateral vehicle handling behaviour. The models were build to accommodate the limitations of the experimental set up. During the building of vehicle control systems, most studies rely on a linear tire models due to its proficiency at simulating low side slip manoeuvres. A non-linear Magic formula model was chosen for this study with goals in mind to produce an accurate model for a large range of handling behaviour.

Deterministic algorithms are widely used by researchers aiming to estimate tire model parameters. This requires an initial guess of the tire parameters and while this method might be accurate while optimizing models using tire force measurements, optimizing with vehicle body accelerations suffer from inaccuracies due to model simplification or measurement noise. This led to the idea that stochastic algorithms might perform better considering its global approach.

Experimental tests were performed with a test vehicle to acquire measurement data required for the optimization process. An extra validation dataset consisting of the return journey from the test location, was recorded to further validate the simulation model containing the parameter sets produced by the algorithms. This dataset reflects normal vehicle operation on public roads which is ultimately is the focus of this thesis and automotive engineering.

Four different algorithms were implemented to optimize the vehicle models. This includes Differential evolution for Multi-objective optimization (DEMO) algorithm, which uses Temporal Difference Q Learning (TDQL) as a method of adapting the algorithm parameters during the optimization process. A generic multi-objective genetic algorithm, Multiple Objective Genetic Algorithm (MOGA) was used alongside to investigate if the adaptive component improved the performance of DEMO. The Goal Attainment algorithm was implemented

along with a simple Mixed Least Squares (MLSQ) algorithm which are deterministic algorithms. They different as the goal attain algorithm is a multi-objective algorithm while the MLSQ algorithm optimizes the average of the two objectives.

The four algorithms DEMO, MOGA, Goal attain and MLSQ were run repeatedly using the 10 datasets available. Standardization were carried out to ensure that the results were comparable such as a common random seed for the initial population for the stochastic algorithms and initial parameter set for the deterministic algorithms. Initial parameters which were varied 20% of the individual parameters bounded range with respect to the initial parameter set chosen for this thesis based on similar tire files and prior knowledge of the Magic formula tire model.

6-2 Conclusion

The results obtained from the algorithm runs shed much light on the nature of the problem investigated by this thesis. The sensitivity analysis of the initial parameter, which is used by deterministic algorithms, show that a small change in starting position can affect the outcome of the optimization. This conclusion plays an important role in the answering of the main research question laid out at the beginning of this thesis.

> Does an adaptive genetic algorithm perform better at estimating tire model parameters than gradient-based algorithms which use pre-defined initial tire model parameters?

It would seem that the results presented in chapter 5 do not point at a straightforward answer to this question. When it comes to achieving the least Normalized Root Mean Squared Deviation (NRMSD) for both objectives, the DEMO algorithm performs statistically better than the deterministic Goal attain algorithm and the non-adaptive MOGA but did not manage to completely outperform the MLSQ algorithm. However, when the computational duration is taken into account, the DEMO algorithm does converge to a solution in less time. DEMO was however not the fastest on average and was statistically outperformed by the MOGA which brings doubt to the requirement of an adaptive component for smaller models and datasets.

Both DEMO and MOGA supply a set of solutions belonging to a Pareto set. This was done to cater to the requirement of the model to optimize two objectives. The Goal attain algorithm was set up to do the same but did not perform well in all aspects of the analysis and is not recommended as a method for this application due to the cascading error. As there has not been a situation where a critical decision needed to be made between two Pareto fronts. It is concluded that a combined objective, such as that of the MLSQ is sufficient. This is also due to the low reliability of the cascading goal attainment algorithm design.

When analysing the parameter set of the step manoeuvre solutions, a large variance was observed of the road friction parameters. This was the opposite of the steady-state test which produced slightly greater consistency in parameter values yet did not perform well when confronted with the step manoeuvre datasets. This leads to the conclusion that a better set of training dataset may be required to produce flexible, yet accurate, model parameters.

Lastly, the models using the optimized parameter solutions were used to simulate the validation dataset. This is a large recording the vehicle state and inputs. It was recorded while driving on Dutch public roads and therefore it contains various driving situations. This is relevant to what developers of automated control system want to simulate. Analysing the simulation data, it was visually concluded that the MOGA algorithm produced that best average result.

The final conclusion of this study is that the DEMO algorithm does perform better than a deterministic algorithm. Although not statistically significant for the objective value, it does perform better in the areas of reliability and computational duration. Compared to the MOGA this study concludes that while the DEMO does produce better objective results, the MOGA performs better when it comes to computational duration and the flexibility of the results. Using the current vehicle models and test data, using the MOGA algorithm would be sufficient.

6-3 Recommendation

Having finished this study, a few recommendations can be made for those seeking to work with these algorithms in relation to vehicle or tire models.

- Increase the complexity of the vehicle model and the scope of the experimental data. It increases the accuracy of the model and the flexibility of its application. Instead of a bicycle model, 4 wheel models with more parameters to estimate will provide a challenge to any optimization algorithm. This is where the DEMO algorithm will have more opportunities to show its abilities. Not only in the realizing its objectives but also performing at a faster rate than other algorithms that slow down with an increase in dimension.
- When restricted to vehicle body inertial values, it is recommended to further investigate test manoeuvres that will reduce the noise and uncertainty when it comes to parameter estimation. A better test will lead to a clearer picture of the location of the global optimum.
- The TDQL machine learning component of the DEMO algorithm is an efficient and powerful tool that was only applied to one algorithm parameter. It would be interesting to see how this sort of machine learning can further be implemented in genetic algorithms. The reward and penalty setting of this method can also be investigated to improve the current results.

Appendix A

Theory

A-1 Complete Magic formula

A tire can operate under an infinite number of conditions and while it is easy to assume that these remain constant, this leads to inaccuracies and will invalidate the model when a significant change has taken place [2]. To compensate for these changes, the coefficients of the magic formula can be modified to vary with changes that include camber, tire load, inflation pressure and tire deflection during driving. How these characteristics affect the coefficients of the model are not known and must also be investigated before the model is considered valid across different situations. The coefficients of these modifications can only be determined when experimental data covering these characteristics are present, which can later be included in the model optimization process. For this thesis, only tire load can accurately be calculated from measured acceleration data. The coefficient names of the following equations are taken from [2] but are based on the equations used by Netherlands Organisation for Applied Scientific Research (TNO) in their manual [33]. Investigating this more advanced model is important as the parameters used are considered a standard within the automotive industry. When model optimization takes place, it is important to have a good indication of similar tire parameters to produce accurate models. Further explanation can be found in section 4-4 covering deterministic algorithms.

$$dF_z = \frac{F_z - F_{zo}}{F_{zo}} \tag{A-1}$$

Equation (A-1) produces the normalized tire load change dF_z where F_z is the tire load and F_{z0} is the nominal tire load.

Without the modification with respect to tire load , the magic formula is assumes that tire load is constant at the nominal tire load , which is the load on which the current coefficients are determined. An coefficient is added to the magic formula coefficients that determine the effect of the normalized tire load change dF_z on the tire characteristics.

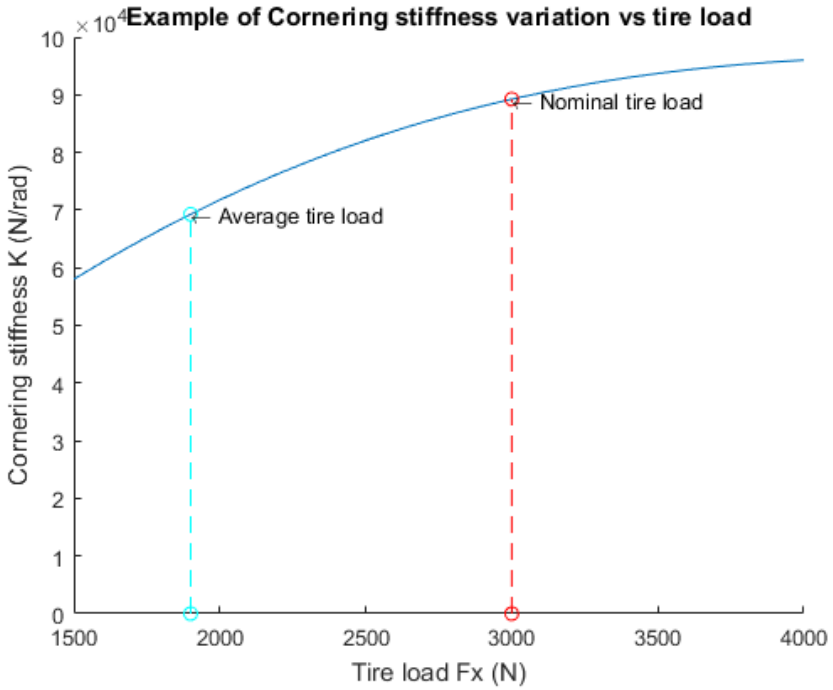


Figure A-1: Variation of cornering stiffness with respect to tire load

$$D = \mu F_z \tag{A-2}$$
$$\mu = p_{Dy1} + p_{Dy2} dF_z \tag{A-3}$$
$$E = p_{Ey1} + p_{Ey2} dF_z \tag{A-4}$$
$$S_{Hy} = p_{Hy1} + p_{Hy2} dF_z \tag{A-5}$$
$$S_{Vy} = p_{Vy1} + p_{Vy2} dF_z \tag{A-6}$$

$$K = p_{Ky1} F_{z0} \sin(p_{Ky4} \arctan(\frac{\frac{F_z}{F_{z0}}}{p_{Ky2}})) \tag{A-7a}$$
$$B = \frac{K}{CD} \tag{A-7b}$$

Equations A-2,A-4 and A-7b show how the change in tire load is taken into account. These are equations taken from [2] and are simplified by assuming that tire air pressure change and camber angle is equal to zero. The primary coefficients D and E are split to include constants, p_{Dy1} and p_{Ey1}, and variables, p_{Dy2} and p_{Ey2}, that is a function of the normalized load change and a constant factor.

Coefficient B is calculated using equation A-7b, which comprises two other primary coefficients as well as K , which is the cornering stiffnessat low slip values. Equation A-7a which determines the values of K contains three coefficients where p_{Ky1} is the maximum values of the coefficient at maximum cornering stiffness. The remainder of the equation is added

to take changes in load into account. p_{Ky2} is the tire load where the maximum cornering stiffnessoccurs while p_{Ky4} is the curvature of the curve which can be seen in figure A-1.

The coefficients presented here are the same that are used in tire information files used by the industry. The Magic formula has also been implemented in a programs called Delft tire which is developed by TNO which is a company working closely with the Delft University of Technology (TU Delft).

A-2 Newton-Gauss method

Most deterministic algorithms use partial derivatives of the model output data with respect to the varied model parameters called the search gradient. Newton and Gauss developed the non-linear least mean squares which uses this gradient within an expanded Taylor series to approximate the algorithms next iteration point. As the partial derivatives of the data are not available, this needs to be calculated using a finite difference method which varies each parameter with a small step δ_{fd} and calculates the difference which produces a value for the Jacobian matrix $\mathbf{J}_{ij}$ shown in equation A-8.

$$J_{ij} = \frac{\partial f_i}{\partial x_j} = \frac{f_i(x + \delta_{fd}) - f_i(x)}{\delta_{fd}} \tag{A-8}$$

where i is the data output and j refers to a model parameter in the vector $\mathbf{X}$.

$$f_{k+1}(\mathbf{X} + \delta_{algo}) = f_k(\mathbf{X}) + (\mathbf{J}_k \delta_{algo}) \tag{A-9}$$

The Jacobian can now be used as part of a Taylor series, shown in (A-9), to compute $f_{k+1}(\delta_{algo})$ which is the approximated model output of the next iteration $k+1$ where k represents the iteration number. As the Taylor series is truncated to one degree, this produces a linear approximation of search space surrounding the current parameter set. The new set of simulation outputs can then be substituted into equation 4-1b to produce the new sum of squared residuals with respect to δ_{algo}. As the goal of the optimization is to find the minimum of this equation, the derivative $\frac{\partial S}{\partial \delta_{algo}}$ is set to zero which produces equation (A-10).

$$\delta_{algo} = (\mathbf{J}^T \mathbf{J}) \mathbf{J} \sum_{i=1}^{n} (y_i - f_i(\mathbf{x})) \tag{A-10}$$

This can be used to solve for δ_{algo} which is the value that is added to the current parameter vector to produce the next iterations parameter vector. At this point the algorithms restarts by determining the new Jacobian matrix for that point [34].

A-3 Genetic operators

There can be many operators that can be applied to a genetic algorithm. The mutation operator is one that comes in many variations that influence the behaviour of the algorithm.

For example a operator that includes the best individual will be more exploitative. An operator that only includes random individuals will be the opposite and explore the search space more.

$$\mathbf{V}_i(t) = \mathbf{X}_{best}(t) + F_1'(\mathbf{X}_{rand2}(t) - \mathbf{X}_{rand1}(t)) \tag{A-11}$$

Equation A-11 is an example of the former. It produces a point along a vector given by the difference between two random points within the search boundaries. The factor F_1 determines how close that point is to the best solution of that generation and is therefore used to control how far the algorithm searches. This operator does not take the current population individual into account and therefore all solutions revolve around the best individual which can lead to the operator being greedy. The validity rate when it comes to the bounds for this mutation operator is 67 %. This means that a third of the time, the operator will have to repeat to find a valid point within the bounds.

Appendix B

Detailed results

Table B-1 contains the weight measurements taken at the weighing station.

Table B-1: Axel weight information for an empty vehicle and including the mass of two adults

	Front (kg)	Rear (kg)	Total (kg)
Empty	860	570	1430
Occupied	950	640	1590

'Source'	'SS'	'df'	'MS'	'Chi-sq'	'Prob>Chi-sq'
'Columns'	[380.9458]	[3]	[126.9819]	[228.5675]	[2.8218e-49]
'Error'	[2.0191e+03]	[1437]	[1.4050]	[]	[]
'Total'	[2400]	[1919]	[]	[]	[]

Figure B-1: ANOVA table for the lateral acceleration NRMSD Friedman test

'Source'	'SS'	'df'	'MS'	'Chi-sq'	'Prob>Chi-sq'
'Columns'	[259.5542]	[3]	[86.5181]	[155.7325]	[1.5276e-33]
'Error'	[2.1404e+03]	[1437]	[1.4895]	[]	[]
'Total'	[2.4000e+03]	[1919]	[]	[]	[]

Figure B-2: ANOVA table for the yaw-rate NRMSD Friedman test

'Source'	'SS'	'df'	'MS'	'Chi-sq'	'Prob>Chi-sq'
'Columns'	[314.9958]	[3]	[104.9986]	[188.9975]	[1.0050e-40]
'Error'	[2.0850e+03]	[1437]	[1.4509]	[]	[]
'Total'	[2400]	[1919]	[]	[]	[]

Figure B-3: ANOVA table for the NRMSD norm Friedman test

'Source'	'SS'	'df'	'MS'	'Chi-sq'	'Prob>Chi-sq'
'Columns'	[1788]	[3]	[596]	[1.0728e+03]	[2.8976e-232]
'Error'	[612]	[1437]	[0.4259]	[]	[]
'Total'	[2400]	[1919]	[]	[]	[]

Figure B-4: ANOVA table for the Computational duration Friedman test

Table B-2: Minimum and Maximum NRMSD norms for the Step manoeuvre runs

	min ObjDist	**max ObjDist**	**min Time**	**max Time**
1 1 GaM	0.125	0.228	22	68
1 1 Goalattain	0.123	0.506	75	177
1 1 Demo	0.123	0.171	17	104
1 1 MixedSVO	0.124	0.201	36	104
1 2 GaM	0.119	0.196	22	52
1 2 Goalattain	0.118	1.177	57	259
1 2 Demo	0.119	0.160	18	73
1 2 MixedSVO	0.120	0.162	22	123
1 3 GaM	0.103	0.142	22	57
1 3 Goalattain	0.106	0.868	49	171
1 3 Demo	0.104	0.127	17	65
1 3 MixedSVO	0.107	0.132	16	109
2 1 GaM	0.122	0.216	22	69
2 1 Goalattain	0.128	1.695	50	225
2 1 Demo	0.116	0.159	20	91
2 1 MixedSVO	0.117	0.841	6	130
2 2 GaM	0.105	0.153	21	58
2 2 Goalattain	0.102	1.318	54	148
2 2 Demo	0.105	0.133	21	63
2 2 MixedSVO	0.104	0.119	25	102
2 3 GaM	0.122	0.161	20	52
2 3 Goalattain	0.120	1.198	59	191
2 3 Demo	0.119	0.144	17	60
2 3 MixedSVO	0.123	0.827	17	132

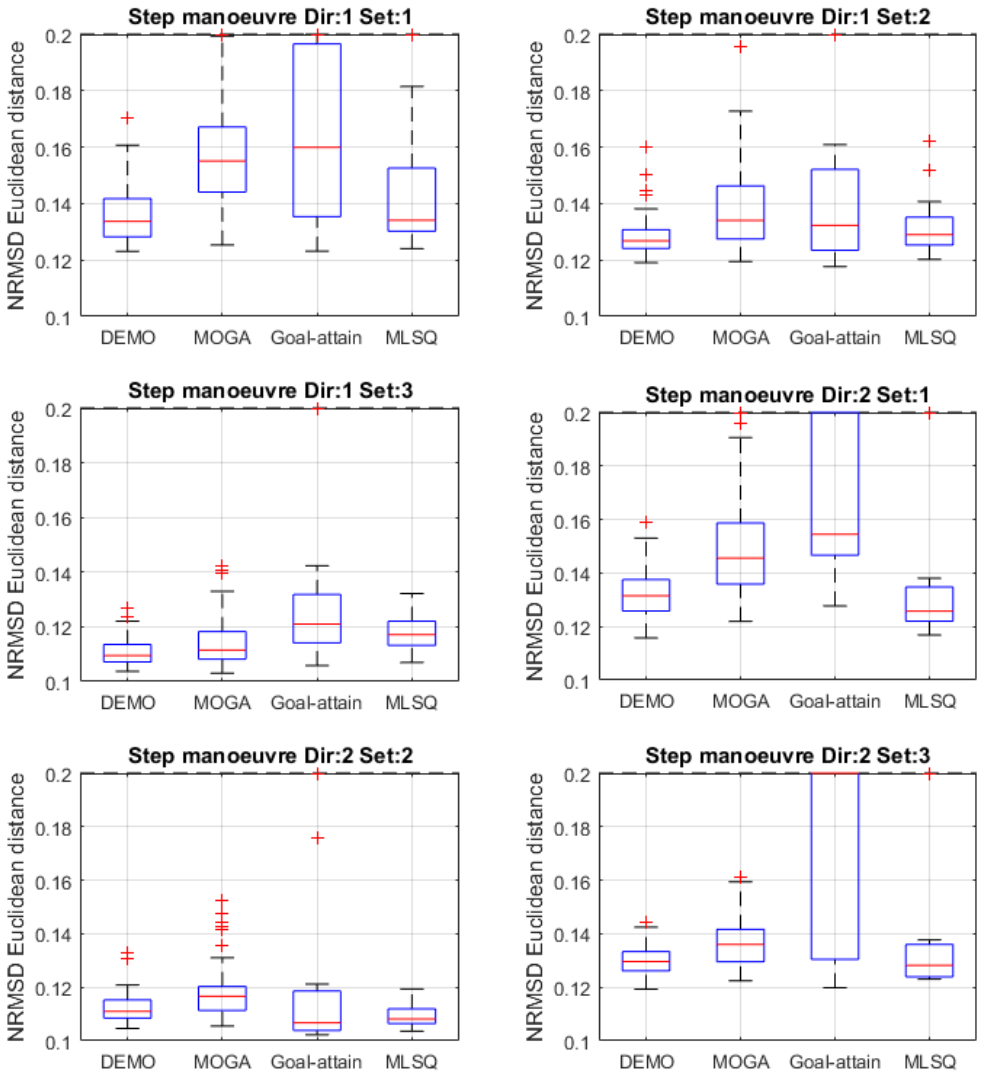


Figure B-5: Box-plot of the individual datasets illustrating the objective value results of the Step manoeuvre

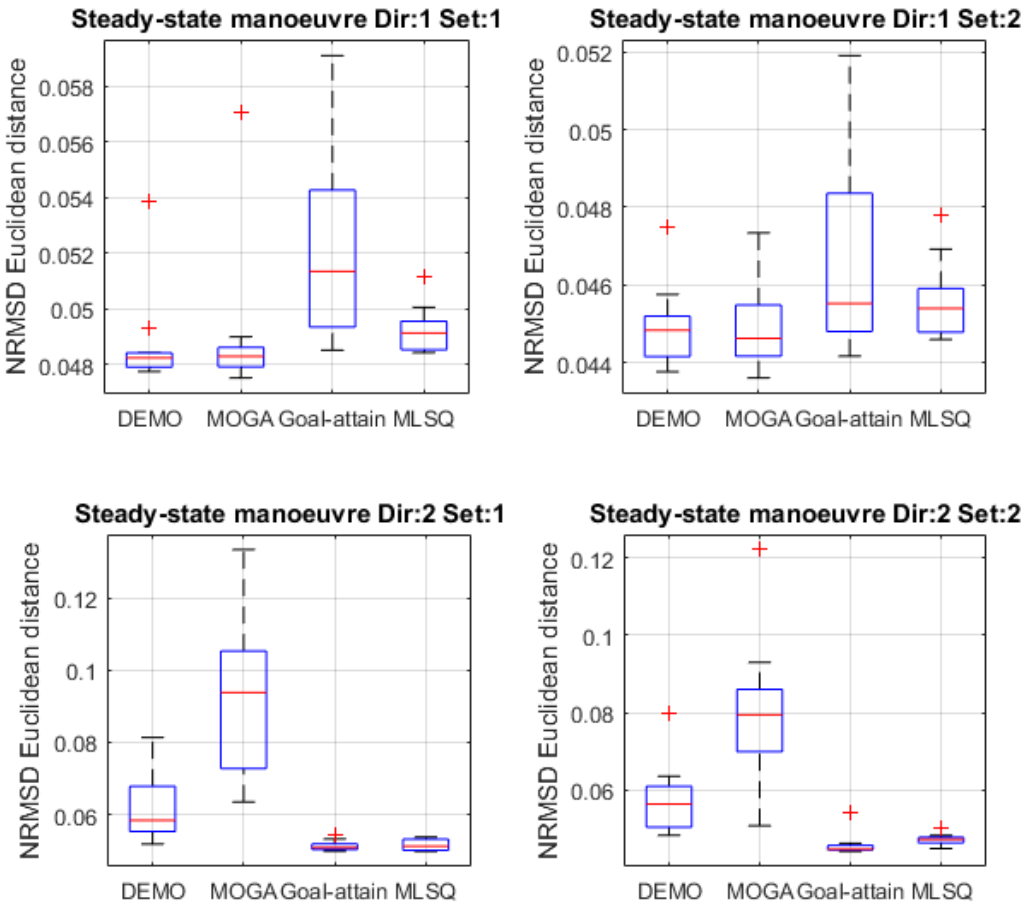


Figure B-6: Box-plot of the individual datasets illustrating the objective value results of the Steady-state manoeuvre

Table B-3: Minimum and Maximum NRMSD norms for the Steady-state manoeuvre runs

	min ObjDist	max ObjDist	min Time	max Time
1 1 GaM	0.048	0.057	47	62
1 1 Goalattain	0.049	0.059	95	167
1 1 Demo	0.048	0.054	37	63
1 1 MixedSVO	0.048	0.051	23	49
1 2 GaM	0.044	0.047	49	68
1 2 Goalattain	0.044	0.052	107	172
1 2 Demo	0.044	0.047	38	66
1 2 MixedSVO	0.045	0.048	28	45
2 1 GaM	0.064	0.133	54	198
2 1 Goalattain	0.050	0.054	123	331
2 1 Demo	0.052	0.081	72	134
2 1 MixedSVO	0.050	0.054	61	281
2 2 GaM	0.051	0.122	63	120
2 2 Goalattain	0.044	0.054	155	455
2 2 Demo	0.048	0.080	55	189
2 2 MixedSVO	0.045	0.050	70	246

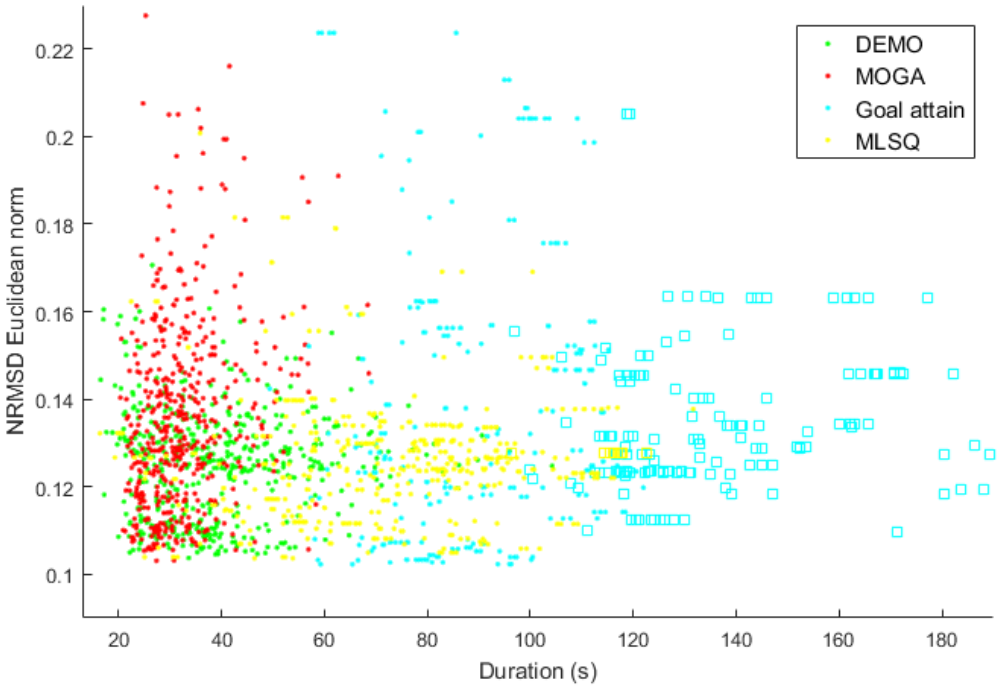


Figure B-7: A scatter plot illustrating the individual results with respect to objective value and computational duration. Square markers indicate that the algorithm reached the maximum number of function evaluations and was terminated on that basis

B-1 Validation

The validation dataset was recorded on the return journey from the test site back to the university. Figure B-8 and B-9 show two intervals that resemble the conditions during the measurement testing.

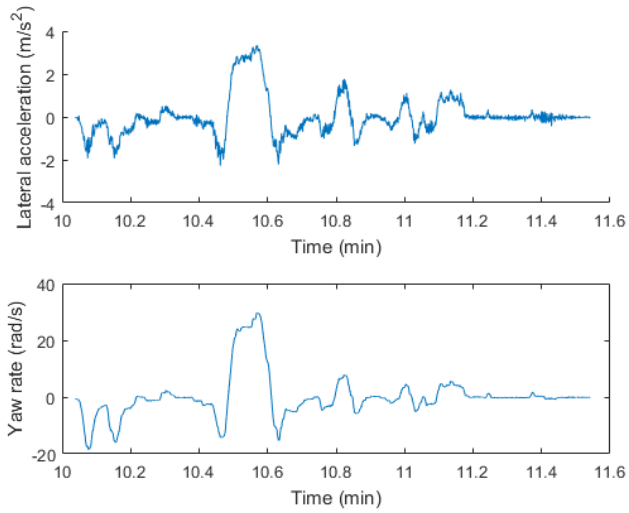


Figure B-8: Plot of a validation data-set interval with an average velocity of $30\,\mathrm{km\,h^{-1}}$

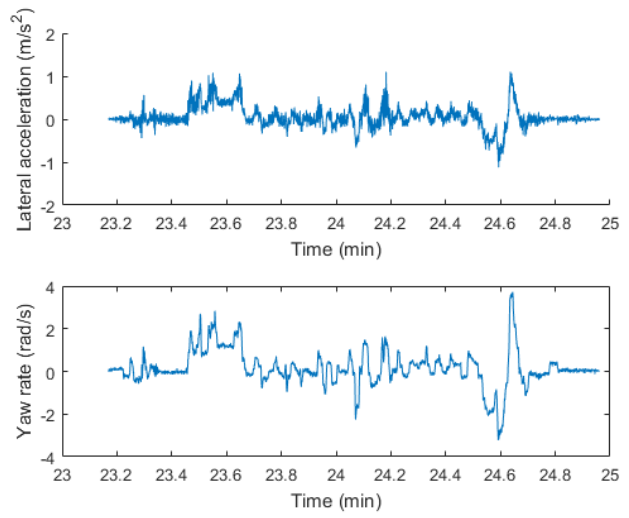


Figure B-9: Plot of a validation data-set interval with an average velocity of $50\,\mathrm{km\,h^{-1}}$

Table B-4 shows the results of the validation dataset using the parameter results produced by the steady-state algorithm.

Table B-4: Objective values of validation test set simulated with results produced by steady-state test set

		DEMO		**GaM**		**Goal att.**		**CLMSQ**	
Dataset	**Interval**	LA	YR	LA	YR	LA	YR	LA	YR
Left 1	1	0.2338	0.0228	0.1927	0.0192	0.2368	0.0230	0.2589	0.0249
	2	0.2958	0.0194	0.1942	0.0132	0.2932	0.0193	0.3394	0.0221
	3	0.3510	0.0208	0.2333	0.0142	0.3518	0.0208	0.4050	0.0238
	4	0.2871	0.0167	0.1788	0.0104	0.2834	0.0164	0.3324	0.0192
	5	0.6393	0.0259	0.4319	0.0181	0.6415	0.0259	0.7370	0.0295
Left 2	1	0.2290	0.0226	0.1927	0.0193	0.2351	0.0230	0.2252	0.0220
	2	0.2874	0.0190	0.1936	0.0132	0.2826	0.0187	0.2652	0.0176
	3	0.3430	0.0204	0.2315	0.0141	0.3403	0.0202	0.3182	0.0190
	4	0.2779	0.0162	0.1785	0.0104	0.2716	0.0158	0.2539	0.0147
	5	0.6273	0.0255	0.4299	0.0181	0.6207	0.0252	0.5812	0.0237
Right 1	1	0.2887	0.0283	0.1607	0.0162	0.2871	0.0280	0.3003	0.0292
	2	0.5351	0.0337	0.1912	0.0125	0.5154	0.0326	0.5487	0.0346
	3	0.5435	0.0317	0.1890	0.0116	0.5235	0.0306	0.5584	0.0326
	4	0.5756	0.0332	0.1935	0.0114	0.5511	0.0318	0.5885	0.0339
	5	0.9242	0.0357	0.3156	0.0132	0.8787	0.0340	0.9485	0.0367
Right 2	1	0.2762	0.0268	0.2052	0.0208	0.3087	0.0296	0.3005	0.0292
	2	0.4829	0.0305	0.3485	0.0221	0.5405	0.0341	0.5394	0.0340
	3	0.4886	0.0286	0.3479	0.0205	0.5494	0.0320	0.5486	0.0320
	4	0.5158	0.0298	0.3715	0.0215	0.5798	0.0333	0.5790	0.0333
	5	0.8151	0.0316	0.5661	0.0221	0.9347	0.0362	0.9341	0.0361

Bibliography

[1] R. Rajamani, *Vehicle Dynamics and Control*. Springer US, 2012.

[2] H. Pacejka, *Tire and Vehicle Dynamics*. Elsevier Science, 2012.

[3] M. Abe, *Vehicle Handling Dynamics*. Elsevier Science & Technology, 2015.

[4] K. C. Dey, A. Rayamajhi, M. Chowdhury, P. Bhavsar, and J. Martin, "Vehicle-to-vehicle (v2v) and vehicle-to-infrastructure (v2i) communication in a heterogeneous wireless network – performance evaluation," *Transportation Research Part C: Emerging Technologies*, vol. 68, pp. 168–184, jul 2016.

[5] S. R. M. D. D. S. Goran S. Vorotovic, Branislav B. Rakicevic, "Determination of cornering stiffness through integration of a mathematical model and real vehicle exploitation parameters," *FME Transactions*, 2013.

[6] D. L. Conte, "Tyre parameter identification from road tests on a complete vehicle," Master's thesis, TU Delft, 2010.

[7] J. V. Ginkel, "Estimating the tire-road friction coefficient based on tire force measurements," Master's thesis, TU Delft, 2014. uuid:3d5ee83c-a35e-4141-8408-ab3916ca1c8a.

[8] V. P. Simon Johansson, "Tire/road friction estimation for front wheel driven vehicle," Master's thesis, LUND University, 2015.

[9] European Conference on Computational Mechanics Solids, *First order tire dynamics*, (Lisbon, Portugal), 2006.

[10] Toyota, Yountville, California, *2010 Toyota Prius Media Preview*, Feb. 2009.

[11] C. A. Ogaja, *Applied GPS for Engineers and Project Managers*. American Society of Civil Engineers, 2011.

[12] D. Katzourakis, J. C. de Winter, S. de Groot, and R. Happee, "Driving simulator parameterization using double-lane change steering metrics as recorded on five modern cars," *Simulation Modelling Practice and Theory*, vol. 26, pp. 96–112, aug 2012.

[13] ISO, "Passenger cars – Steady-state circular driving behaviour – Open-loop test methods," standard, International Organization for Standardization, Geneva, CH, 2004.

[14] ISO, "Road vehicles – Lateral transient response test methods – Open-loop test methods," standard, International Organization for Standardization, Geneva, CH, Feb. 2003.

[15] L. Verhoeff, "Bmw mftyre file 225 50r17." TNO tire file, 2005.

[16] P. Ngatchou, A. Zarei, and A. El-Sharkawi, "Pareto multi objective optimization," in *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pp. 84–91, Nov 2005.

[17] "Matlab documentation - mathworks benelux." https://nl.mathworks.com/help/matlab/. (Accessed on 11/23/2017).

[18] *Evolutionary Multi-Criterion Optimization*. Springer Berlin Heidelberg, 2001.

[19] Y.-x. Yuan, "Recent advances in trust region algorithms," *Mathematical Programming*, vol. 151, pp. 249–281, Jun 2015.

[20] F. V. Berghen, "Levenberg-marquardt algorithms vs trust region algorithms," 2004. UniversitÂt̊e Libre de Bruxelles.

[21] R. Brayton, S. Director, G. Hachtel, and L. Vidigal, "A new algorithm for statistical circuit design based on quasi-newton methods and function splitting," *IEEE Transactions on Circuits and Systems*, vol. 26, pp. 784–794, sep 1979.

[22] J. Snyman, *Practical Mathematical Optimization: 97 (Applied Optimization)*. Springer US, 2005.

[23] S. N. Deepa and S. N. Sivanandam, *Introduction to Genetic Algorithms*. Springer, 2010.

[24] A. Ortiz, J. A. Cabrera, A. Guerra, and A. Simon, "The IMMa optimisation algorithm without control input parameters," *Vehicle System Dynamics*, vol. 47, pp. 243–264, feb 2009.

[25] L. Busoniu, R. Babuska, B. D. Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators (Automation and Control Engineering)*. CRC Press, 2010.

[26] P. Rakshit, A. Konar, E. Kim, and A. K. Nagar, "DEMO-TDQL: An adaptive multi-objective optimization algorithm," in *2013 IEEE Congress on Evolutionary Computation*, IEEE, jun 2013.

[27] F.-A. Fortin and M. Parizeau, "Revisiting the NSGA-II crowding-distance computation," in *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO 13*, ACM Press, 2013.

[28] J. Zhang and A. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 945–958, oct 2009.

[29] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L. C. Jain, and A. K. Nagar, "Realization of an adaptive memetic algorithm using differential evolution and q-learning: A case study in multirobot path planning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, pp. 814–831, jul 2013.

[30] G. Rangaiah, S. Sharma, and H. Lin, "Evaluation of two termination criteria in evolutionary algorithms for multi-objective optimization of complex chemical processes," *Chemical Engineering Research and Design*, vol. 124, pp. 58–65, aug 2017.

[31] B. Calvo and G. Santafé, "scmamp: Statistical comparison of multiple algorithms in multiple problems," 2015.

[32] K. M. Ramachandran and C. P. Tsokos, *Mathematical Statistics with Applications in R, Second Edition*. Academic Press, 2014.

[33] T. Automotive, *MF-Tyre/MF-Swift 6.2*. TNO, PO Box 756 5700 AT Helmond The Netherlands, 5/1/2014.

[34] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

Glossary

List of Acronyms

TU Delft	Delft University of Technology
TNO	Netherlands Organisation for Applied Scientific Research
RMSD	Root Mean Squared Deviation
NRMSD	Normalized Root Mean Squared Deviation
GNSS	Global Navigational Satellite System
INS	Inertial Navigation System
RTK	Real-time Kinematic System
GPS	Global Positioning System
ECU	Engine Control Unit
CAN	Controller Area Network
ISO	International Organization of Standardization
DEMO	Differential evolution for Multi-objective optimization
TDQL	Temporal Difference Q Learning
SQP	Sequential Quadratic Programming
EPS	Electronically assisted Power Steering
KKT	Karush Kuhn Tucker
BFGS	Broyden Fletcher Goldfarb Shanno
MOGA	Multiple Objective Genetic Algorithm
MLSQ	Mixed Least Squares

List of Symbols

α_f	Front slip angle
α_r	Rear slip Angle
$\boldsymbol{\lambda}$	Lagrangian bounds multiplier
δ_{fd}	Finite difference step change
δ_{steer}	Wheel steering angle
κ	Slip ratio
$\dot{\Psi}$	Yaw rate
α	Learning rate
$\ddot{\Psi}$	Change in vehicle yaw rate
$\ddot{y}$	Lateral acceleration produced by motion along the y-axis
$\Delta S(\mathbf{X})$	Gradient vector
$\hat{y}_i$	Average of the experimental data
$\mathbf{d}_k$	Newton step in the direction of the local solution
$\mathbf{H}$	Hessian matrix
$\mathbf{J}_{ij}$	Jacobian matrix
$\mathbf{Q}_{reward}(t)$	Reward table
$\mathbf{R}_i$	Mutated population
$\mathbf{V}$	Mutated individual
$\mathbf{X}$	Parameter vector
$\mathcal{B}$	Bound constraint matrix
$\mathcal{F}_i^*$	Objective goals
$\mathcal{L}$	Lagrangian equation
ρ_0	Steady-state cornering radius
τ_y	Relaxation length lag coefficient
a_x	Longitudinal acceleration
a_y	Inertial acceleration experienced by the centre of gravity
B	Stiffness factor
C	Shape factor
CD_i	Pareto front
Cr	Crossover probability
D	Peak value
dF_z	Normalized vertical load change
E	Curvature factor
$f(x_i)$	Simulation data point
F_j	Mutation coefficient
F_y^D	Dynamic lateral force
F_y^S	Steady-state lateral force output of the tire model
F_z	Normal tire load

F_{z0}	Nominal tire load
GD_m	Generational distance
h_c	Height of the centre of gravity
I_z	Vehicle moment of inertia
K	Cornering stiffness
$K_{penalty}(t)$	Penalty value
l	Length of the wheelbase of the vehicle
m	Total mass of the vehicle
$M(\mathbf{d})$	Approximation simulation residuals produced by $\mathbf{X}_k$
P	Centre of gravity
p_{Dy1}	Friction coefficient
p_{Dy2}	Variation of the Friction coefficient with load
p_{Ey1}	Curvature coefficient
p_{Ey2}	Variation of the curvature coefficient with load
p_{Ky1}	Maximum values of coefficient B at maximum cornering stiffness
p_{Ky2}	Load where maximum cornering stiffness occurs
p_{Ky4}	Curvature of K vs Fz
PF_i	Pareto front
$R_i(\mathbf{x})$	Residual
RL	Relaxation length
$S(\mathbf{x})$	Residual squared
S_h	Horizontal shift
S_v	Vertical shift
$U_i(t)$	Candidate population
v_{cp}	Contact patch velocity
w_i	Goal attain objective weight
y_i	Experimental data point
F_x	Longitudinal tire force
F_y	Lateral tire force velocity
F_z	Normal tire load
l_f	Distance to the front axle from the centre of gravity
l_r	Distance to the rear axle from the centre of gravity
M_z	Aligning torque
v_y	Lateral velocity