DEGREE PROJECT IN VEHICLE ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

PARAMETER OPTIMIZATION OF EPAS USING CAE

SHOUNAK BHATTACHARYYA

SURAJ SIVARAMAKRISHNAN

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Master of Science Thesis

Parameter Optimisation of EPAS Using CAE

Shounak Bhattacharyya

Suraj Sivaramakrishnan

Approved	Examiner	Supervisor
29-06-2018	Mikael Nybacka	Marcus Ljungberg
	Commissioner	Contact Person
	Volvo Cars Corporation	Hans Backstrom

Sammanfattning

För att upprätthålla ett positivt momentum i såväl tekniska som logistiska utmaningar på dagens bilmarknad har stora biltillverkare börjat använda sig av virtuella simuleringsverktyg. Dessa verktyg möjliggör utveckling av diverse fordonsmodeller långt innan resurser investeras i en fysisk prototyp.

Detta projekt fokuserar på utvecklingen av ett verktyg som potentiellt kan hjälpa att optimera dynamiska beteendeparametrar för ett fordon. Detta uppnås genom att skapa en optimeringsrutin för att ställa in de olika parametrarna för den elektroniska servostyrningen (EPAS). Denna process görs vanligtvis manuellt, genom test på provbana, på grund av svårigheterna att korrelera subjektiva bedömningar (SA) med objektiva mätetal (OM). Att automatisera denna process kan bidra till att minska den övergripande forsknings- och utvecklingstiden genom att tillhandahålla en baslinje för EPAS-parametrarna som i efterhand kan finjusteras genom manuell justering på provbana.

Verktyget är byggt genom att ansluta olika program i en optimeringsmiljö som kallas ModeFrontier. Modellering och simuleringar utförs i IPG CarMaker, med efterbehandling av resultaten i Sympathy for Data. Flera optimeringsalgoritmer testades för att uppnå bästa optimeringsrutinen. EPAS-parametrarna består av det grundläggande styrmomentet, aktiv retur och aktiv dämpning, och fungerar som invärden till optimeringsrutinen där utvärdena från modellen är objektiva mätetalen, vilket ger en tydlig indikation på den dynamiska prestandan hos en komponent. Dessa mätvärden optimeras för att passa Steering DNA-strukturen, som unikt beskriver egenskaperna hos ett fordon. Det slutliga optimerade fordonet testas manuellt på provbana för att bestämma den verkliga körkänslan.

Abstract

To keep up with technological as well as logistical challenges of the modern automobile market, major car manufacturing firms have resorted to virtual simulation tools. This enables the development as well as validation of vehicular models much before resources are invested into a new physical prototype.

This project focuses on the development of a tool that would help in optimising the handling parameters of a vehicle. This is achieved by creating an optimization routine for tuning the various parameters of the Electronic Power Steering (EPAS). This process is usually done manually, by on-track testing, due to the difficulties in correlating Subjective Assessments (SA) with Objective Metrics (OM). Automating this process would help to reduce the overall research and development time, by providing a baseline tune for the EPAS parameters which could then be finely tweaked by manual track testing.

The tool is built by interfacing various software in a multi-objective optimisation environment known as ModeFrontier. The modelling and simulations are performed in IPG CarMaker, with the post processing of the results taken care of by Sympathy for Data. Multiple optimisation algorithms were tested to achieve the best optimisation routine. The EPAS parameters, namely the Basic Steering Torque, Active Return and Active Damping, act as the input to the optimization routine. The outputs of the model are the Objective Metrics, which provide a clear indication of the dynamic performance of a component. These metrics are optimised to fit the Steering DNA structure, which uniquely describes the attributes of a vehicle. The final optimised vehicle is manually tested at the track, to determine the real driving feel.

Keywords: Electronic Power Assist Steering, Optimization, Subjective Assessment, Objective Metrics

Acknowledgement

We would like to thank Dr. Mikael Nybacka, Associate Professor at the Division of Vehicle Dynamics at KTH, and Marcus Ljungberg, Vehicle Dynamics Engineer at Volvo Cars Corporation for being our thesis supervisors and supporting us through the course of the thesis. A special thanks to Alejandro Gonzalez, Eshwar Sondhi and Axel Jonson from the Vehicle Dynamics Group at Volvo Cars for all their help and support during the course of the project. We would also like to thank Hans Bäckström, Manager of Vehicle Dynamics at Volvo Cars and Egbert Bakker, Technical Leader of Vehicle Dynamics at Volvo Cars for their guidance.

Tack så mycket

Shounak Bhattacharyya & Suraj Sivaramakrishnan
KTH Royal Institute of Technology
Stockholm, Sweden

Nomenclature

ABBREVIATIONS

AD	Active Damping
AR	Active Return
BST	Basic Steering Torque
DOE	Design of Experiments
ECU	Electronic Control Unit
$EPAS$	Electronic Power Assisted Steering
FMU	Functional Mock-Up Unit
$MOGA$	Multi-Objective Genetic Algorithm
OM	Objective metrics
SA	Subjective Assessments
SiL	Software In the Loop

Contents

1. Introduction

This master thesis was carried out at the Driving Dynamics department in **Volvo Car Corporation, Gothenburg**. Volvo Cars was founded in 1927 as a subsidiary of the ball bearing manufacturer SKF. Volvo Cars manufactures and markets vehicles of various types including sport utility vehicles, station wagons and executive sedans.

In a competitive market with rapid development of technology, vehicle manufacturers are moving their research and development work towards a virtual, simulation driven environment. Cars are getting more complex, with customers having greater demands, expecting releases of multiple variants in short spans of time. With conventional manufacturing and testing methods, these demands are difficult to adhere to, and hence the switch to virtual development was imminent. With a reduction in the number of physical prototypes required, manufacturers are able to reduce costs, lead times, and stress on resources. This method of development is very beneficial to both manufacturers and the environment.

Reduction of lead time is of high priority in the automotive industry, and this includes reducing time for development. When dealing with vehicle testing for handling and steering feel, a new vehicle concept requires extensive testing to achieve a base tune, before further fine tuning of the vehicle, to deem it production ready. This is a fairly long process, and also occurs relatively later in the design phase, and requires data from vehicles of previous generations. The objective of the thesis was to remove these dependencies, by optimising the electronic power steering parameters to achieve an ideal base tune for the vehicle.

Vehicle dynamics testing in general is a particularly tricky domain to move into the virtual environment, since a large portion of the physical testing is concentrated on the feeling and experience of driving a car. Gómez et al. (2015) found correlations between these subjective feelings and the objective test metrics, which will be used in the virtual environment. [1] The advantage of dealing with objective metrics is that the final assessments are not dependant on the individual carrying out the tests.

In order to isolate the objective metrics and extract the ideal results from testing, the vehicles are subjected to different maneuvers. These maneuvers help to highlight a certain aspect of the

handling performance of the vehicle. A validated virtual vehicle model is used, and multiple iterations of the various maneuvers are performed to help obtain an optimal model. All these simulations can be performed on various software, but its a challenge to compile the results into something substantial. Hence, this optimal model is obtained with the help of an optimisation software which invokes different optimisation routines. A well-defined optimization routine on one single platform rather than multiple software's ensures the physical labour, as well as the time required for manual tuning of the model real-time on the track, is reduced.

2. Literature Study

2.1 EPAS System

The steering subsystem forms the basis for any interaction between the driver and the wheels. The most simple and commonly used method to implement this is the rack and pinion steering. As the name suggests, the rotational movement of the steering wheel is converted to the translational movement of the rack with the help of a pinion gear. The rack is further connected to the tie-rods which aid in changing the directional motion of the wheels. However, this mechanical system requires heavy input from the driver, especially at lower speeds, to negotiate sharp turns and hence is not considered an ergonomically viable solution in premium vehicles. Thus, in order to improve the handling and stability of the vehicle, keeping in mind the driving experience, the Electronic Power Assist Steering, commonly known as the EPAS, was developed. The primary objective of this system was to provide an assisting torque to the driver, thus reducing the required driver input. [2]

There exist multiple ways through which the power assist can be induced in the steering. These include hydraulic systems, electro-hydraulic systems, and the EPAS. The most popular amongst these is the EPAS, where an assisting torque is provided on the steering column or an assisting force on the rack. The EPAS comes in the following variants.

1. Motor mounted the steering column.
2. Motor mounted on the steering rack connected co-axially via a belt and ball nut gear.
3. Motor driving a second pinion gear.

The steering system analyzed during the course of this project came in the second variant, with a belt and ball nut gear, as seen in the figure 2.1. The EPAS was then modelled in a fully functional vehicle environment for further evaluation and optimization. [3]

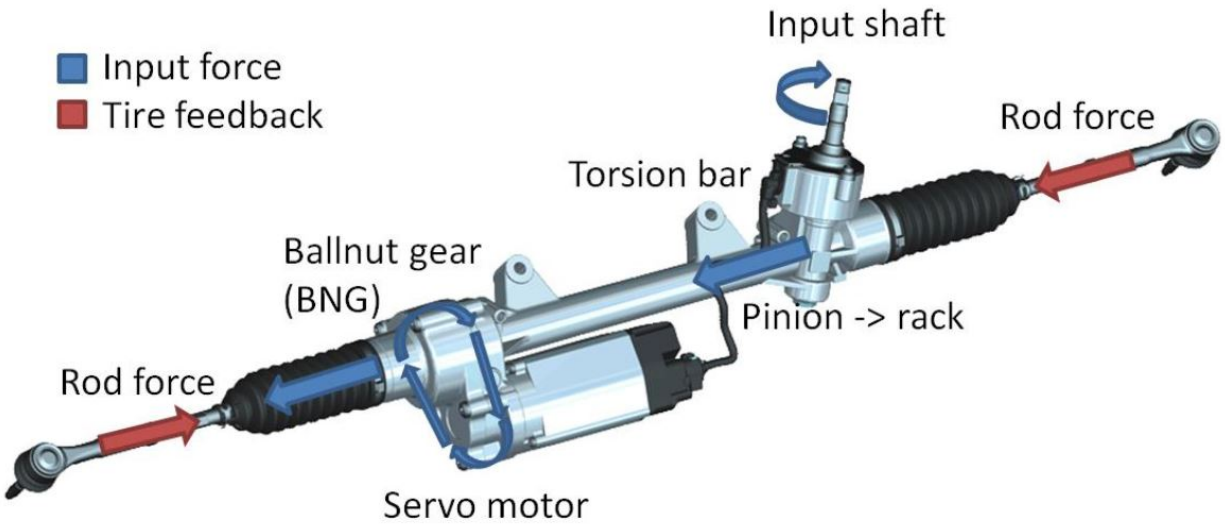


Figure 2.1: Electronic Power Assist Steering

2.1.1 Modelling

The steering subsystem is a software-in-loop (SiL) model, designed in the Simulink environment, thus having the ability to be accessed by IPG CarMaker for full vehicle analyses via script files coded in MATLAB. As mentioned earlier, the EPAS system consisted of two distinct subsystems, the electronics and the mechanics. This was replicated in the Simulink model for the steering, thus allowing easier access to parameters for identification and modification purposes. [3]

- **Mechanics**

  The mechanical part of the model represents the transmission of the external driver input from the steering column through the torsion bar, and the pinion gear to the rack. Other major elements include the modelling of the friction elements, the definition of inertia's for the various rotating masses, and the modelling of the servo, rack, and pinion gears.

  Each sub-block consists of equations representing the inputs, outputs, and the functions present using mathematical operations. Parameters such as angles, angular velocities, and torque were used to define these equations. The primary inputs to the components such as the spindle, upper steering column, input shaft, torsion bar, output shaft, and the rack were well documented.

  Each of these components have a non-linear, speed dependent friction model comprising static, tangent-hyperbolic and hysteresis frictions. These frictional models also depend on the aforementioned parameters. Furthermore, the models also consist of a combination of

look-up tables and constants which are variant specific.

The blocks within the Simulink interface are formulated on the basis of mathematical equations which are used to define the model, thus accurately depicting the functioning of steering system.

- **Electronics**

  The electronics part of the model mainly comprises of the servo motor, the ball nut gear, and the Electronic Control Unit (ECU). These blocks comprise of multiple input and output signals corresponding to parameters such as the voltage, current, and temperature of the ECU. The ECU block also consists of signals which refer to control parameters, acting on multiple subsystems of the vehicle model.

  Most functions of the SiL steering model are stored within the electronic block and form the core functionalities of the steering system. Additionally, the parameters from the mechanical subsystem are also referenced as they work in tandem, in order to produce the desired level of assistance necessary for the driver.

2.1.2 Functions

Within the EPAS model in the Simulink interface, there exist multiple functions ranging from Steering system controller settings to torque creating functions. [3] The parameters which act as the inputs to the optimization process are primarily the torque creating functions in the steering system. They influence certain attributes of the EPAS, such as the assistance torque provided, the return-ability of the steering, the friction compensation in the system, the end stops for the rack travel as seen in figure 2.2.

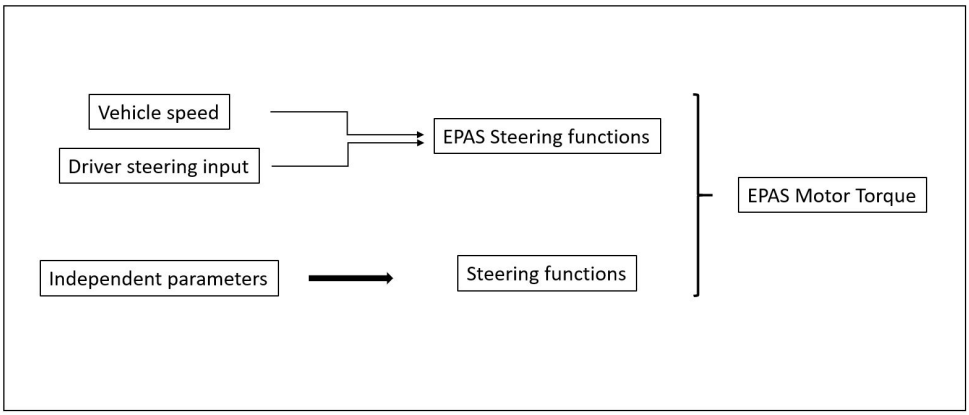


Figure 2.2: Torque creating functions

A summation of these functions result in the generation of the motor torque, which serves the

purpose of the EPAS system. These built-in functions consist of parameters which are enclosed within the ECU block of the Simulink interface and are elaborated below.

- **Basic steering torque**

  The first function dealt with was the basic steering torque, more commonly known as the boost curves. Primary parameters involved in the definition of this function are the rack assist forces from the steering and the longitudinal velocity of the vehicle. The boost curves are formulated on the basis of the servotronic functionality of the EPAS system, where assistance is provided with the help of the motor either hydraulically or electrically. [4]

  The values of the steering torque were plotted as a function of the rack assist force provided to the system. The basic steering torque function was logarithmic in nature, which monotonously increased with an increment in speed.

  An increase in speed resulted in an increased level of assistance torque provided for the same values of rack assist force. The values of assistance torque saturated beyond a certain value, for increasing values of rack assist force. Thus, manipulating these boost curves would give an indication of how heavy or light the steering felt under different driving conditions. Another important aspect which could be studied from these curves was the on-center behaviour of the steering. [4]

- **Active return**

  The second function which was studied was the active return. As the name suggests, this function controls the return-ability of the steering wheel, for any input provided by the driver. Hence the primary parameter influencing the return was the magnitude of the steering wheel angle. As mentioned previously, most functions within the steer subsystem are control signals. The ECU thus acts as a multiple input - multiple out (MIMO) control block. This results in the possibility of multiple steering functions being coupled. The return-ability function in this case coupled with the damping function is influential in limiting to motor assist torque output. Computing and modifying the return-ability function is useful as it is indicative of the metrics pertaining to the overshoot and return behaviour of the steering. [3]

- **Active damping**

  The active damping function, as the name suggests, affects the damping of the steering input. Coupled to the active return function, it essentially has effects on both the return-ability as well as maintaining the stability of the steering under different driving conditions.

The Damping function specifically aids in the stabilization of the rotor under transients. These conditions involve highly dynamic steering inputs.

The magnitude of most control functions present within the steering system are limited by their characteristic curves and extended boundary conditions. The values of parameters such as the return speed, assist torque, and rack assist force cannot exceed certain pre-defined values. Meeting these conditions preserved the functionalities and the effects they had on the overall driving performance. In addition to the overshoot and return behaviour of the steering, the active damping function is also indicative of the metrics corresponding to steering torque feedback. [2]

2.1.3 Tuning catalogue

Each of the torque creating functions have multiple parameters stored within the configuration file (.xml), which influence the final motor assist torque acting on the system. In order to identify the variables or parameters which actually influence the magnitude of these functions, a tuning catalogue published by the supplier of the model was used. Thus, a list of parameters that can be calibrated to achieve optimal results was defined.

- **Basic steering torque**

  In the case of the assistance function, the calibration parameters are a list of 7 boost curves which are defined at vehicle velocities between parking speeds and highway driving speeds as seen in figure 2.3. [2]

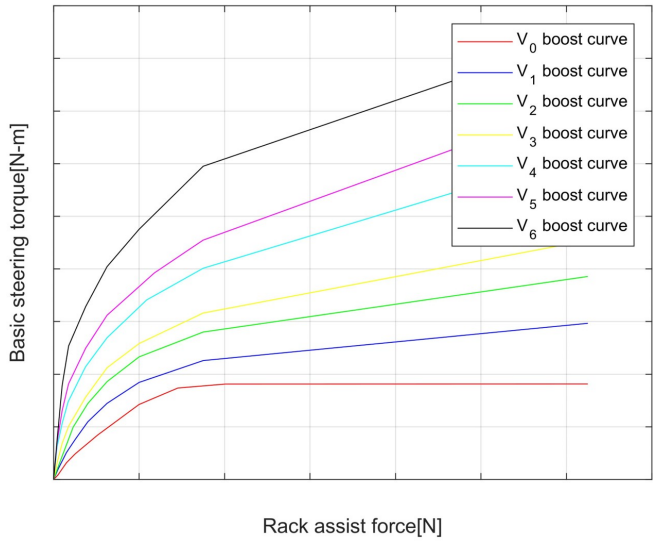


Figure 2.3: Basic Steering Torque

- **Active return**

  For the return function the identified calibration parameters are dependent on the inputs provided by the driver, which affect the return behaviour of the steering under different driving condition (longitudinal velocities).

- **Active damping**

  In order to calibrate the damping function, four parameters which can be influenced by driver input are considered. Each of these parameters are plotted as a function of the factor of active damping present in the system.

2.2 IPG CarMaker

The primary simulation environment for the purpose of this project was IPG CarMaker. Due to its ability to define maneuvers, evaluate vehicle dynamics characteristics, pick tire models, and complete simulations five times faster than real time, CarMaker helps reducing the lead time on the completion of the optimization routine. The CarMaker user interface can be seen in figure 2.4.

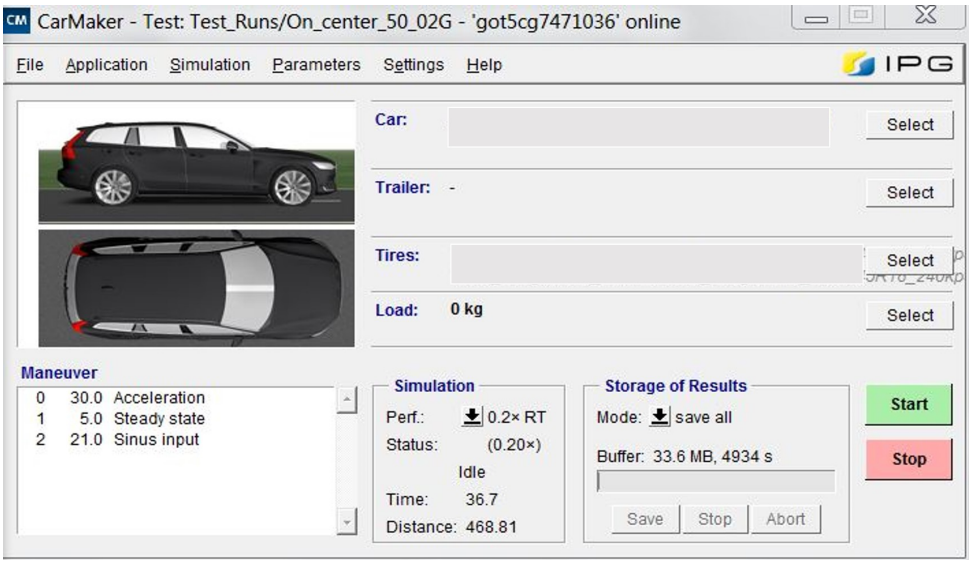


Figure 2.4: IPG CarMaker

An influential feature which set CarMaker apart from other multi-body simulation environments was its ability to replace any subsystem in the vehicle with a custom model defined by the user. Since CarMaker interfaces really well with both MATLAB and Simulink, the customized software in loop model could be plugged in. The results from these simulations could then

further be exported to multiple post processing environments vis-à-vis Matlab, Sympathy for Data, and ADAMS Car for analysis.

2.3 Optimisation Software

ModeFrontier is a comprehensive multidisciplinary and multi-objective optimization software. Its innovative algorithms and effective integration with leading simulation tools ease the engineering process. ModeFrontier has become essential for increasing the understanding of cost and performance factors while reducing the product development time in multiple industries. [5]

ModeFrontier does away with traditional engineering practices of finding the optimal solution using trial and error, and instead employs the concept of intelligent design space exploration using various optimisation algorithms. The software allows users to build up a logic workflow to graphically formulate the problem, followed by evaluating and optimising the designs with the ability to monitor the progress in real-time.

2.4 Uniform Latin Hypercube Sampling

The Latin Hypercube Sampling Method, similar to the Monte Carlo Sampling Method, generates a set of random points between a given set of limits. However, since the Monte Carlo Method relies on pure randomness, it can generate points which aren't uniform over the design space, resulting in crowded areas or areas left unexplored. The Latin Hypercube Sampling Method, however, though relying on randomness as well, conforms to a uniform distribution. Hence the points are more spread out, with nearly the entire design space being explored. This is also very helpful when running a small number of iterations, in comparison to the number of parameters. [6]

2.5 Optimisation algorithms

Many different optimisation algorithms were tried and tested for the project. Finally, two algorithms were decided on, for their varied approaches, but simple implementations. The two algorithms were Simplex and MOGA2.

2.5.1 Simplex

The Simplex Method in ModeFrontier, also known as Downhill Simplex or the Nelder Mead Method, is a common optimisation algorithm used to obtain the maximum or minimum of a cost function in multi-dimensional space. This heuristic algorithm uses the concept of a simplex, which is a polyhedron with N+1 vertices in an N-dimensional space. The algorithm tries to iteratively alter the worst vertex, bringing the final value closer to that of the ideal design. The algorithm stops when the difference between the final and penultimate design is lower than the termination accuracy, or when the maximum number of iterations is reached. [7]

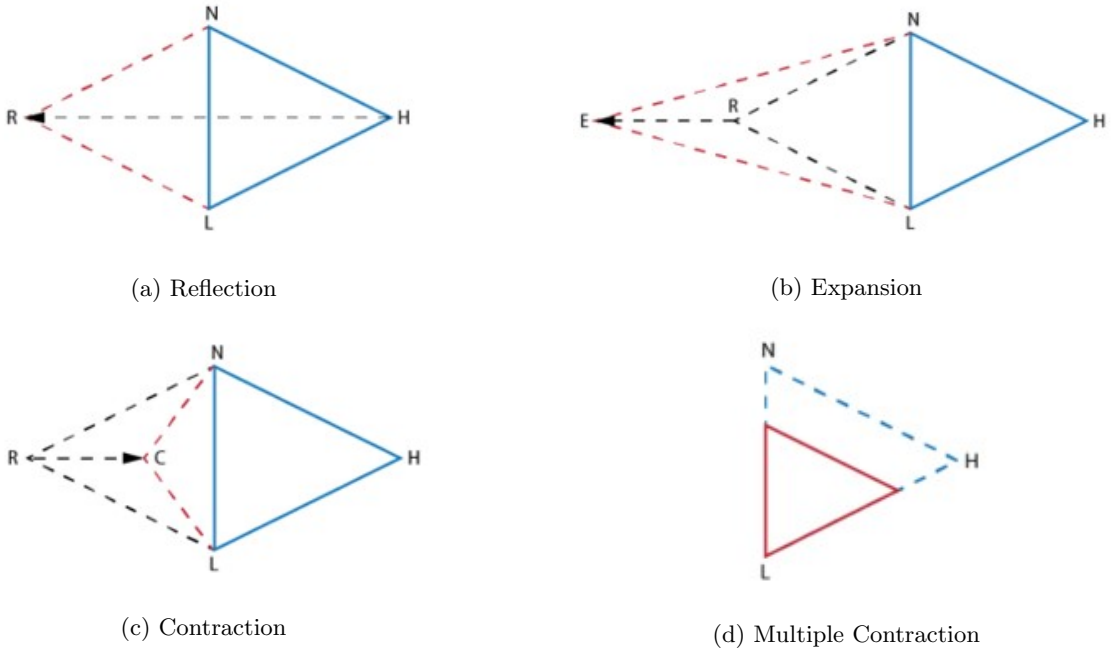
The Downhill Simplex performs four unique operations in order to improve the positioning of the worst vertex. These are,

1. Reflection: The algorithm first tries to reflect the position of the worst vertex in the opposite direction, as shown in figure 2.5a. If the reflected point is equivalent to the best point before the operation, the Simplex is accepted. In case the reflected point is better or worse than the best point before the operation, the Simplex moves onto the consequent operations.

2. Expansion: For the cases where the reflected point is better than the best point before the operation, the reflected point is further expanded in the direction of the reflection, as shown in figure 2.5b. If the point obtained through expansion is not as good as the original reflected point, the algorithm reverts to the original reflection.

3. Contraction: For the cases where the reflected point is worse than the best point before the operation but better than the worst point before the operation, the reflected point is shrunk back against the direction of the optimisation, as shown in figure 2.5c. If this point is better than the original reflected point, the Simplex is accepted, else the algorithm reverts to the original reflection.

4. Multiple Contraction: For the cases where the reflected point is worse than the worst point before the operation, the reflection is discarded and all the points of the Simplex, with the exception of the best point, is contracted towards the best point. This operation is shown in figure 2.5d.



(a) Reflection

(b) Expansion

(c) Contraction

(d) Multiple Contraction

Figure 2.5: Simplex

In figure 2.5, H is the worst point, L is the best point, N is any other point on the polyhedron and R is the reflected point.

2.5.2 Multi-Objective Genetic Algorithm (MOGA-II)

Genetic Algorithm is a type of optimisation method which is based on natural selection and Darwin's Theory of Evolution. MOGA-II is an improved version of the original Multi-Objective Genetic Algorithm by Poloni. It uses the concept of multi-search elitism, which allows it to preserve solutions without prematurely converging into a local optimum and improves the overall convergence of the algorithm. [8]

The elitism operator proceeds in the following manner to try an obtain the global optimum.

- Each of the discrete parameters helps to form a chain, the concatenation of all of which finally forms the chromosomes. The algorithm begins with the initial population of chromosomes and uses it to generate newer generations or offspring. This is done using one of the following operations, chosen at each step and applied to the parent.

  1. Crossover

     (a) One Point Crossover: The interchanging of the parent chromosomes to produce the two offspring is done by randomly selecting a crossover point.

(b) Two Point Crossover: The interchanging of the parent chromosomes to produce the two offspring is done by randomly selecting two crossover points.

(c) Uniform Crossover: The interchange is handled a little differently, with the algorithm deciding which of the parents will contribute which gene to the offspring. This allows the parents to interact at the gene level instead of the segment level. Figure 2.6 shows the different types of crossovers.

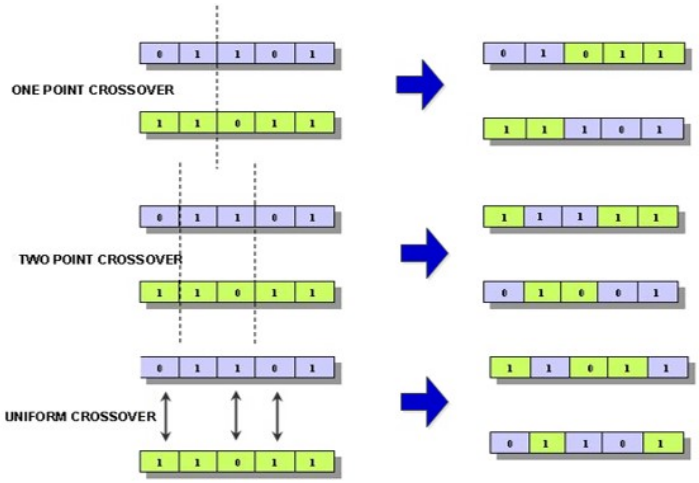


Figure 2.6: MOGA-II

2. Mutation: The genetic material of the chromosome is altered and the new design may be entirely new to the gene pool. Mutation helps in preventing the algorithm from saturating at a local optimum.

- Once all of the offspring are generated, the 'fitness' of the individuals are computed to arrive at the best design. The Darwinian evolutionary theory of Survival of the Fittest is then employed. All the non-dominated designs are added to the elite set, and any duplicate or dominated designs are removed.
- The next generations are then computed and the process is continued till convergence is achieved or the maximum number of generations are reached.

2.6 Subjective Evaluation

In addition to objective tests and optimisation, subjective testing is necessary to gauge the driveability of the final design iteration of any optimisation. Subjective evaluations are primarily done by experienced test driver, who can accurately describe subtle and specific changes in

vehicle behaviour. A major part of the subjective testing and tuning performed at Volvo is done at their test track in Hällered, including the tuning of the EPAS parameters.

The car is tested on the High Speed Oval for on-centre behaviour, and on Handling Track 2 for on-the-limit handling performance, as can be seen in figure 2.7.

Figure 2.7: Volvo Proving Grounds - Hällered

3. Maneuvers and metrics

The influence of both subjective assessments and objective metrics on the study of vehicle performance have already been mentioned in the previous section. The subjective assessments are performed by multiple expert drivers in a single-blind test, where they have minimal information of the vehicle configuration. Following a series of tests which include the first impression, constant radius, slaloms a subjective feedback is obtained from the driver. The feedback is specific to certain dynamic attributes of the vehicle. In most cases the feedback from the drivers are both recorded and filled in on a pre-defined questionnaire. The final feedback is then translated to ratings which can then be used as a tool for comparison between different configurations of the same vehicle as well as comparison between different vehicles. [1]

The objective metrics are functions derived from the dynamic vehicle parameters obtained from similar test events performed by the drivers. The values of these Metrics, have pre-defined ranges based on the kind of configuration deemed necessary by the vehicle manufacturer. The links between SA's and OM's have played a vital role in understanding the behaviour of the vehicle from the perspective of the driver and comparing them against results obtained from the CAE simulations. [1]

3.1 Maneuvers

Any multi-objective optimization routine is heavily dependent on the inputs to the process and the outputs from the process. However, the optimization of any input and its ability to satisfy a particular value for the output depends on the quality of the loadcase defined. Since the focus of the optimization process was on the driving performance, the loadcases used were basically different dynamic and steady state maneuvers. These maneuvers could be defined with multiple variants, with each variant being specific to a certain region in the driving spectrum. These regions could be distinguished and characterized by the values of certain dynamic parameters obtained such as the lateral acceleration, yaw rate. The definition of these variants depended on parameters such as the lateral acceleration and the longitudinal velocity of the vehicle. The

following loadcases were used for the purpose of this project.

1. **Ramp steer**

   The ramp steer maneuver is performed at a specific velocity with a steady increment in Steering wheel Angle over the duration of the manoeuvre as seen in figure 3.1. Factors such as the Lateral Acceleration and Yaw rate at steady state can be obtained from the maneuver. [3]

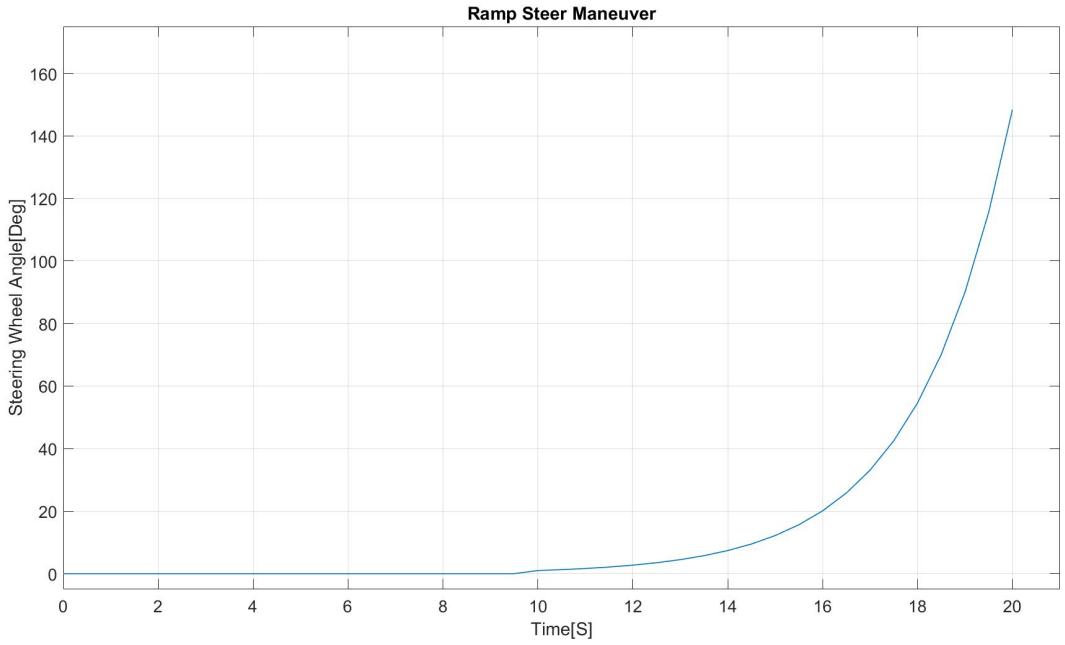


Figure 3.1: Ramp Steer Maneuver

   Since the value of steering wheel angle at a specific lateral acceleration could be obtained from the maneuver, ramp steer was used as a pre-event for most dynamic maneuvers used to determine the dynamic behaviour of the vehicle.

2. **On-Center**

   The on-center test as per definition, characterized the steering performance at low frequencies and moderate lateral accelerations. At values of lateral accelerations that were low, it is indicative of the steering performance with mild corrections. At values of lateral acceleration considered moderate, it is indicative of steering performance with heavy inputs. For instance, negotiating a hair-pin bends. [3]

   As seen in figure 3.2, the maneuver is performed by providing a sinusoidal steering input with a fixed amplitude and frequency. Multiple variations of the on-center test could be

used, depending on the desired response of the vehicle to be examined. These variations include a range of longitudinal velocities at low and moderate lateral accelerations. The frequency of steering input was limited to prevent the saturation of tyres by ensuring they stayed in the linear range.

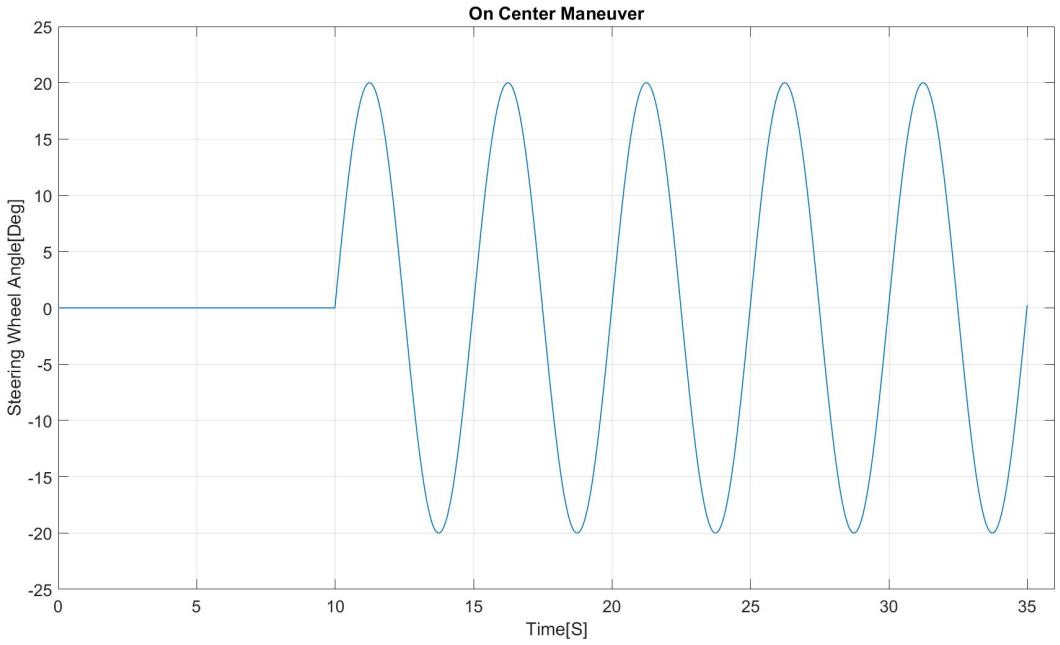


Figure 3.2: On Center Maneuver

The steering wheel angle amplitudes at the specified lateral accelerations were obtained from the pre-event, simulated at the specified speed with a steady state steering input over a duration of time.

3. **Swept steer**

The swept steer maneuver covers a wide range of vehicular characteristics which includes straight ahead drivability, steady state turning as well the lateral dynamics (roll behaviour) thus forming the complete driving spectrum.

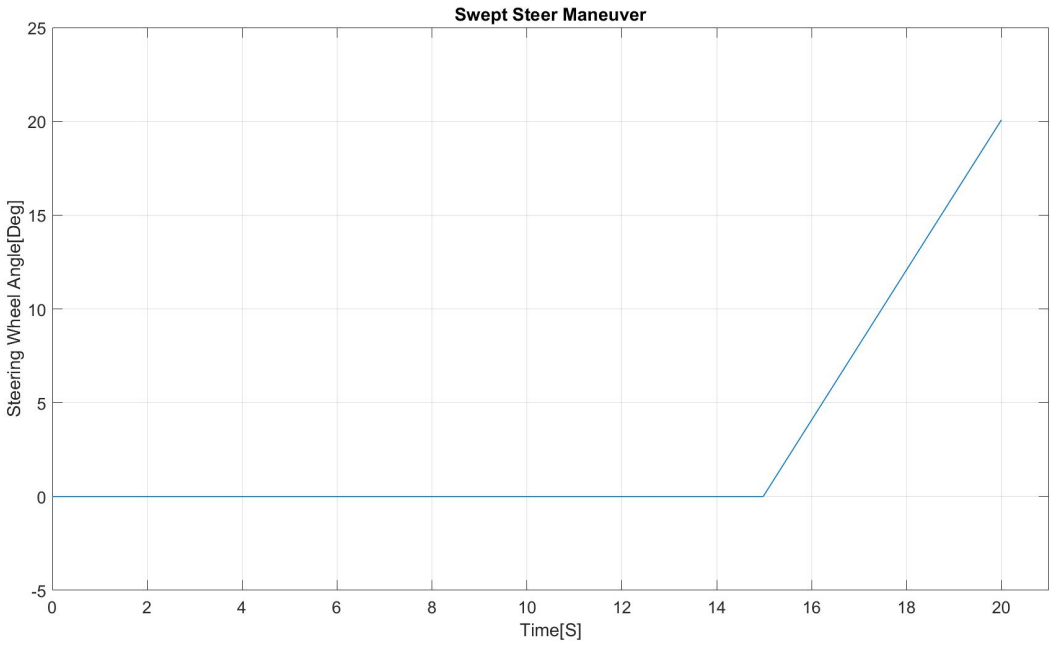


Figure 3.3: Swept Steer Maneuver

From figure 3.3 it can be seen that maneuver was performed by providing a steering input with constant jerks, thereby building up the lateral acceleration upto the desired value. Two variations of the swept steer maneuver were used for the purpose of this project.

(a) Low-G swept steer (LSS)

This particular test-run was performed at high speeds. The methodology for the maneuver was the same, where the steering wheel angle input was provided ensuring the lateral acceleration is built up in steps upto a predefined target value. [3]

The steering wheel angle amplitude was obtained on the basis of the steering wheel velocity (rad/S) required to build up lateral acceleration in the above defined procedure. The Low-G swept steer characterized the straight-ahead drivability of the vehicle subject to mild turns on the steering wheel.

(b) High-G swept steer (HSS)

This particular test-run was performed at a velocity lower than the Low-G swept steer. However for the same velocity, the pre-defined target value of lateral acceleration was higher. [1]

Thus, the High-G swept steer characterized the steady state turning behaviour of the vehicle as well as the roll dynamics at both ends of the driving spectrum.

4. **Frequency response**

The frequency response maneuver characterizes the turning performance of the vehicle at both steady states and transients. In the linear driving range, the magnitude and phase delays of variables such as the lateral acceleration and yaw rate at different steering wheel frequencies were determined. [1]

A frequency sweep is performed at high speeds with a steering wheel angle amplitude corresponding to moderate values of lateral acceleration. The steering wheel angle frequency was gradually increased in steps of 0.1 Hz as seen in figure 3.4.

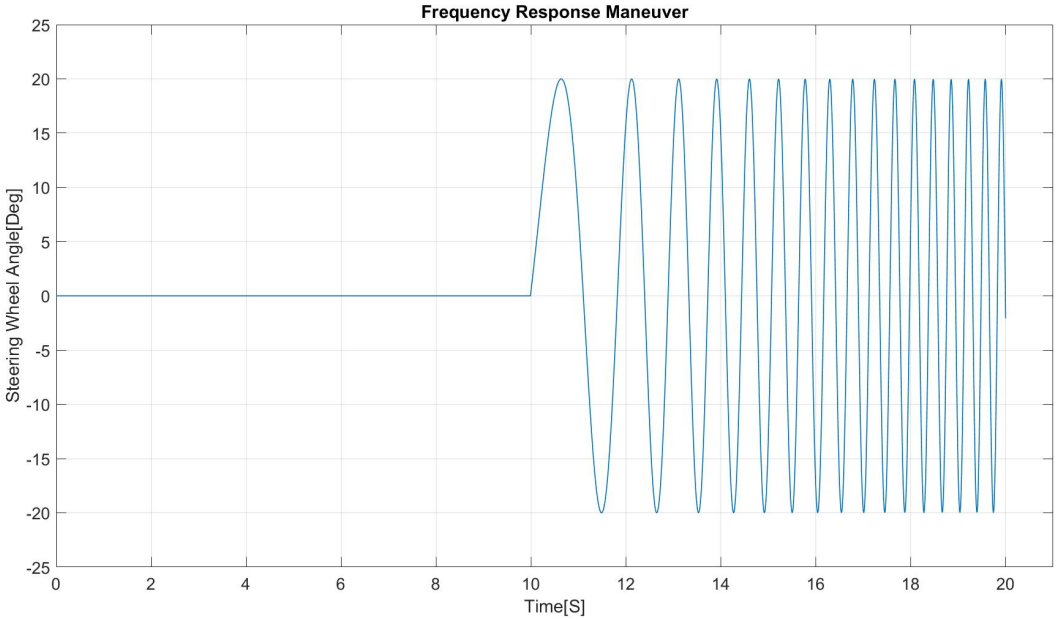


Figure 3.4: Frequency Response Maneuver

5. **Sine with dwell**

The sine with dwell is an evasive manoeuvre which is used to evaluate the vehicle responses under unseen circumstances. It characterizes the time delays between variables such as the yaw rate response and the steering wheel angle amplitude.

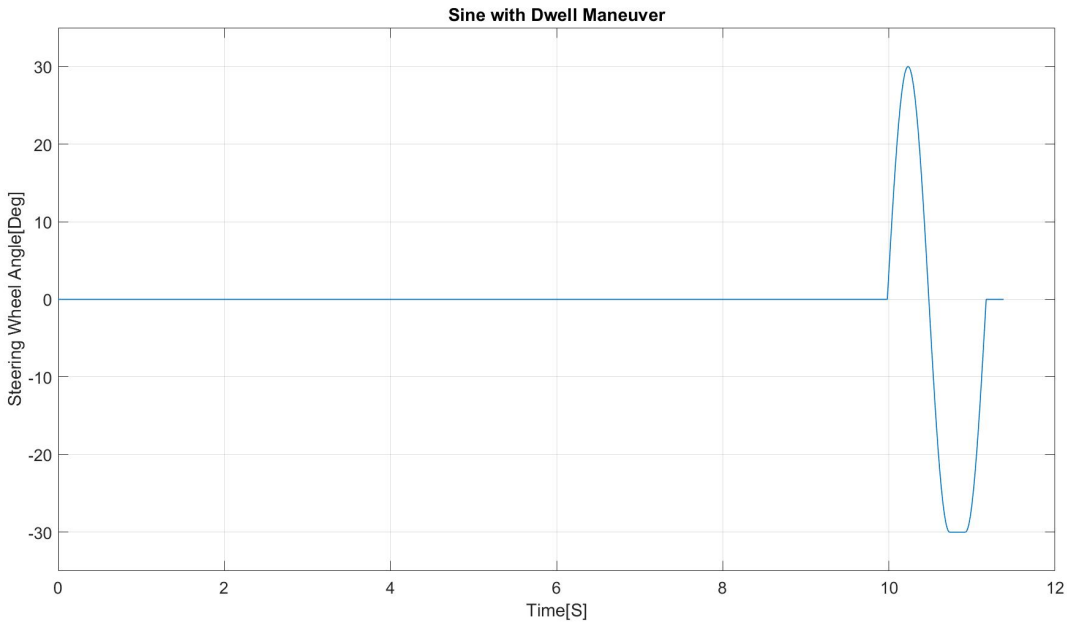


Figure 3.5: Sine With Dwell Maneuver

From figure 3.5 it can be seen that the test-run is performed by giving a sinusoidal input with the steering amplitude being held constant at the peak value for a certain duration.This steering amplitude is obtained from the pre-event. The input is then multiplied by factors in steps of 0.5 until the vehicle rolls over. [1]

3.2 Objective Metrics

As discussed in the previous sections, OM's were functions that were used to quantify in numbers,the dynamic performance of the vehicle, which in turn helped build the DNA. Each of these metrics has a specific target value, within a pre-allocated range. This target was considered to be the optimal value which would enhance the performance of a particular attribute within the subsystem. Furthermore, these metrics could be used as a bench marking tool by car manufacturers to compare, analyze and optimize the performance of any particular subsystem based on the competition in the market.

Since the focus was majorly on the steering system, following a literature survey on the previous work done on the subject, a list of 27 OM's were identified. The list of all the metrics can be seen in figure 3.6. The quantification of the vehicle performance could be broadly classified into 3 levels. [1]

Level 3	Level 4	Level 5	Unit	Measure
Straight-ahead Controllability	Response	Window (SWA at 0.05 g)	[°]	OM-1
		Yaw response gain on-centre	$[°/s/100°SWA]$	OM-2
		Lateral acceleration response gain low speed	$[g/100°SWA]$	OM-3
		Lateral acceleration response gain high speed	$[g/100°SWA]$	OM-4
		Gain linearity (Steering sensitivity ratio)	[−]	OM-5
		Response time delay	$[ms]$	OM-6
	Roll Control	Total roll-rate gradient at 1 Hz (on-centre)	$[°/s/g]$	OM-7
	Torque Feedback	Torque dead-band (SWA at 1.3 Nm)	[°]	OM-8
		Torque build-up (SWA Torsional rate)	$[Nm/100°SWA]$	OM-9
		Friction feel (Torque at 0 g)	$[Nm]$	OM-10
Cornering Controllability	Response	Yaw response gain off-centre	$[°/s/100°SWA]$	OM-11
		Linear range understeer gradient	$[°/g]$	OM-12
		Yaw gain linearity	[%]	OM-13
		Yaw gain at maximum lateral acceleration / maximum yaw gain	$[°/s/100°SWA]$	OM-14
		Yaw - SWA phase time lag at 4m/s2	$[ms]$	OM-15
		Ay - SWA phase time lag at 4m/s2	$[ms]$	OM-16
		Ay - Yaw phase time lag at 4m/s2	$[ms]$	OM-17
	Roll Control	Total roll-rate gradient during cornering	$[°/g]$	OM-18
	Torque Feedback	SWA torque build-up into the corner	$[Nm/100°SWA]$	OM-19
		SWA torque build-up cornering	$[Nm/g]$	OM-20
		On-centre hysteresis (Torque dead-band in degrees)	[°]	OM-21
		Off-centre hysteresis (Torque hysteresis at 0.3 g)	$[Nm]$	OM-22
		Effort level (Torque at 0.3 g)	$[Nm]$	OM-23
First impression	-	Yaw low speed response gain on-centre	$[°/s/100°SWA]$	OM-24
		Low speed torque build-up (maximum SWA torsional rate)	$[Nm/100°SWA]$	OM-25
		Parking effort standstill	$[Nm]$	OM-26
		Parking effort rolling	$[Nm]$	OM-27

Figure 3.6: List of Objective Metrics

The broadest classification included the straight ahead controllability, cornering ability and the first impression tests. At a more dynamic-specific level the metrics were classified on the basis of attributes such as the response, roll control and the torque feedback.

As each of these metrics are maneuver specific, the reliability and accuracy of the final values depend on the event performed. The objective metrics are functions that take into account a fixed number of vehicle parameters. There was a possibility that the response obtained from multiple metrics would be indicative of the same vehicle characteristics. Furthermore, these Metrics were the outputs to the optimization process. Hence, taking all 27 metrics into consideration would result in unreliable results. [1]

Keeping in mind the aforementioned possibilities, a correlation study was performed for all the 27 objective metrics. Following the study as seen in figure 3.7, the Pearson's linear correlation coefficient was obtained. The values of this co-efficient varied between -1 and 1, with values over 0 representing a positive, and the values less than 0 representing a negative correlation.

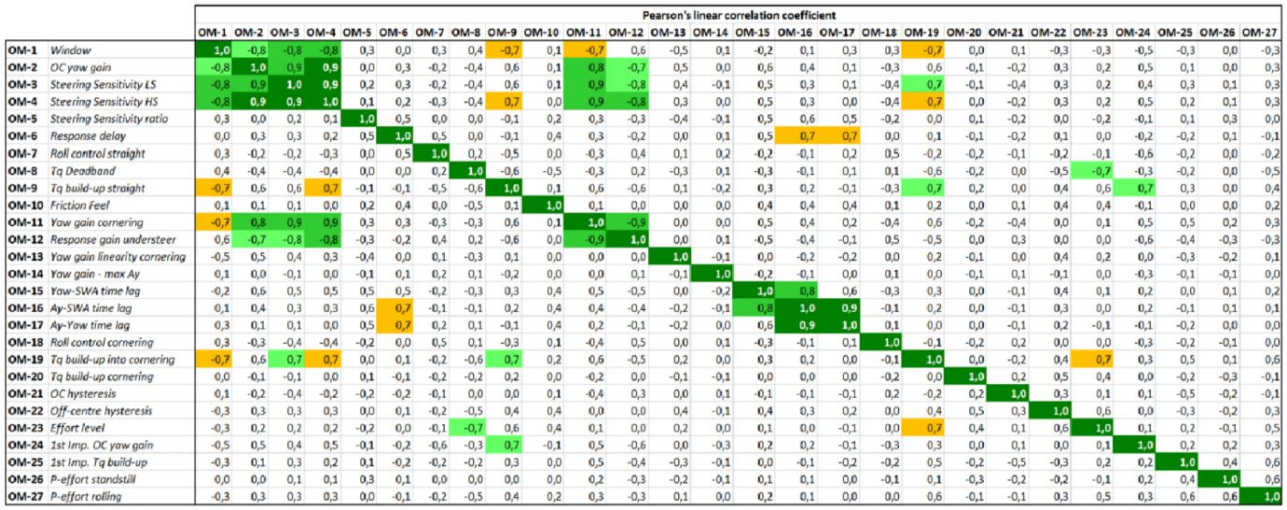


Figure 3.7: correlation study of OM's

Any value greater than 0.5 indicated a strong correlation between the two metrics in question. Similarly any value less than -0.5 indicated a strong negative correlation. On this basis a list of the following 10 OM's were chosen, which covered the dynamic performance of the vehicle under all possible driving conditions. [1]

1. **Lateral acceleration response gain (high speed)**

   This metric represents the steady state value of acceleration gain. It Is indicative of the sensitivity of the steering experienced by the driver at high speeds. The value of this metric could be computed from equation 3.1

$$\text{Response gain} = \frac{A_y}{SWA * 100} \tag{3.1}$$

$$\text{where } A_y = \text{Lateral acceleration in G's,}$$
$$SWA = \text{Steering wheel angle input in Degrees}$$

   Which is the gradient of the cross plot between the lateral acceleration and the steering wheel angle .

2. **Gain linearity**

   The steering sensitivity ratio as the name suggests is a ratio between the value of steering sensitivity between moderate and low values of lateral acceleration. The steering sensitivity

could be computed from the gradient of the cross-plot between the lateral acceleration and the steering wheel angle between the defined values of lateral acceleration normalized to 100°of steering wheel angle.

3. **Torque build-up**

   Also known as torsional rate, this metric represents the steering wheel torque per 100 units of steering wheel angle. Measured in [Nm/100 SWA] (as seen in equation 3.2), it was indicative of the stiffness of the steering wheel felt by the driver.

$$\text{Build-up} = \frac{SWT}{SWA * 100} \tag{3.2}$$

$$\text{where SWT} = \text{Steering Wheel Torque in N-m,}$$
$$\text{SWA} = \text{Steering wheel angle input in } ^\circ$$

   The metric was obtained from the slope of the cross plot between steering wheel angle and steering wheel torque.

4. **Torque dead-band**

   The torque dead-band in degrees refers to the Steering wheel angle at which a specific value of steering wheel torque is produced. Measured in [degrees], it is an average of the left and right turn direction metrics.

   The metric was obtained from the cross plot between the steering wheel angle and steering wheel torque. Similar to the torsional rate, the dead-band is representative of the torque feedback in the vehicle under straight ahead driving with minute changes.

5. **Friction feel**

   The friction feel for the steering could be defined as the measure of friction or damping present in the steering system during on-center. It is an indicator of the mechanical friction present in the system, thus the effort required to overcome the damping present in the steering system.

   The value for friction feel was obtained by taking the average of direction response for left and right turns in the cross-plot for steering wheel angle and lateral acceleration where the value for lateral acceleration is zero.

6. **Lateral acceleration – steering wheel angle phase time lag**

   The phase time lag could be defined as the equivalent time of the frequency when the lateral acceleration lags the steering wheel angle signal by 45°. This can be seen in equation 3.3

$$\text{Time Lag} = \frac{1}{8 * f_{45^\circ}} \tag{3.3}$$

$$\text{where } f_{45^\circ} = \text{Frequency at } 45^\circ\text{phase lag in Hz}$$

   Measured in [mS] this metric is representative of the steering response of the vehicle. A lower value indicated a better response whereas a higher value indicated a worse response.

7. **Total roll-rate gradient during cornering**

   The roll gradient during cornering measured in [deg/G] is representative of the roll dynamics of the vehicle. It characterizes both the steady state as well as the transient driving conditions under cornering.

   In quantitative terms the value of the metric was the slope of the cross-plot between the lateral acceleration and the roll angle which is represented in equation 3.4.

$$\text{Roll rate} = \frac{\text{Roll Angle}}{A_y} \tag{3.4}$$

$$\text{where } A_y = \text{Lateral acceleration}$$
$$\text{Roll Angle} = \text{Roll angle of vehicle during cornering in } ^\circ$$

   Since the metric indicates cornering ability under both steady state and transient driving, a highly dynamic maneuver could be used to determine the value.

3.3 Correlation study

Following the definition of both the inputs (ECU Parameters) and the outputs (Objective Metrics) to the optimization process, one stand-out feature was the dependency on the velocities. On one hand the inputs which were to be optimized were parameters defined as a function

of the vehicular velocity, whereas the outputs which helped quantify the results were metrics which depended on the maneuvers performed at specific vehicular velocities.

Manoeuvre	Metric	Units	Speed
On-Center	Lateral acceleration response gain	g/(100° SWA)	Highway speed
Low-G Swept Steer	Steering sensitivity ratio	-	Highway speed
Low-G Swept Steer	Torque dead band	Deg	Highway speed
Low-G Swept Steer	Torque build-up	Nm/(100° SWA)	Highway speed
On-Center	Friction feel	Nm	Highway speed
High-G Swept Steer	Total roll-rate during cornering	Deg/g	Country road speed
High-G Swept Steer	SWA torque build-up cornering	Nm/g	Country road speed
High-G Swept Steer	Effort level	Nm	Country road speed
Sine-with-Dwell	Ay-SWA Phase time lag at 4 m/s^2	ms	Country road speed
Parking effort	Parking effort rolling	-	Standstill condition

Figure 3.8: List of Maneuvers and OM's

This fact was essential in narrowing down the number of events that were required to produce the necessary results. Thus, from a list of 7 maneuvers which included 18 variants, only 9 loadcases(including variants) were deemed necessary to obtain the desired responses. For instance, since the lateral acceleration response gain metric evaluated the response gain at high speeds with mild turns, the on-center maneuver could be used to obtain an accurate value. Similarly metrics such as the gain linearity, torque build-up and torque dead-band were heavily dependent on the values of steering wheel torque in steady state. Thus, indicative of the steering response under straight driving. Hence the low-G swept steer maneuver could be used to estimate the value of these metrics. In metrics such as roll-rate, required high values of lateral acceleration. Thus, dynamic maneuvers such as the frequency response, or high-G swept steer would provide a more accurate indication of the values of these metrics. The final list of maneuvers used can be seen in figure 3.8. This resulted in a more robust optimization process with a much lower lead time on both the simulation as well as post processing environments.

4. Simulation environment

IPG CarMaker was the primary simulation environment within the optimization loop. The initial setup of the vehicle model on IPG CarMaker involved designing the road surface and defining the type of driver who would execute the maneuver. Furthermore, parameterization of a predefined vehicle model was done using a combination of look-up tables from measurement or multi-body simulation data. These were primarily the kinematics and compliance (K&C) tests for the suspension model in the vehicle. They included values for the stiffness of springs (N/mm) ,dampers (N/mm), bushings and anti-roll bars on both the front and rear ends of the vehicle. These values were obtained from simulations on ADAMS Car a multi-body simulation software from MSC, which specializes in building and analyzing vehicle dynamics models. [9]

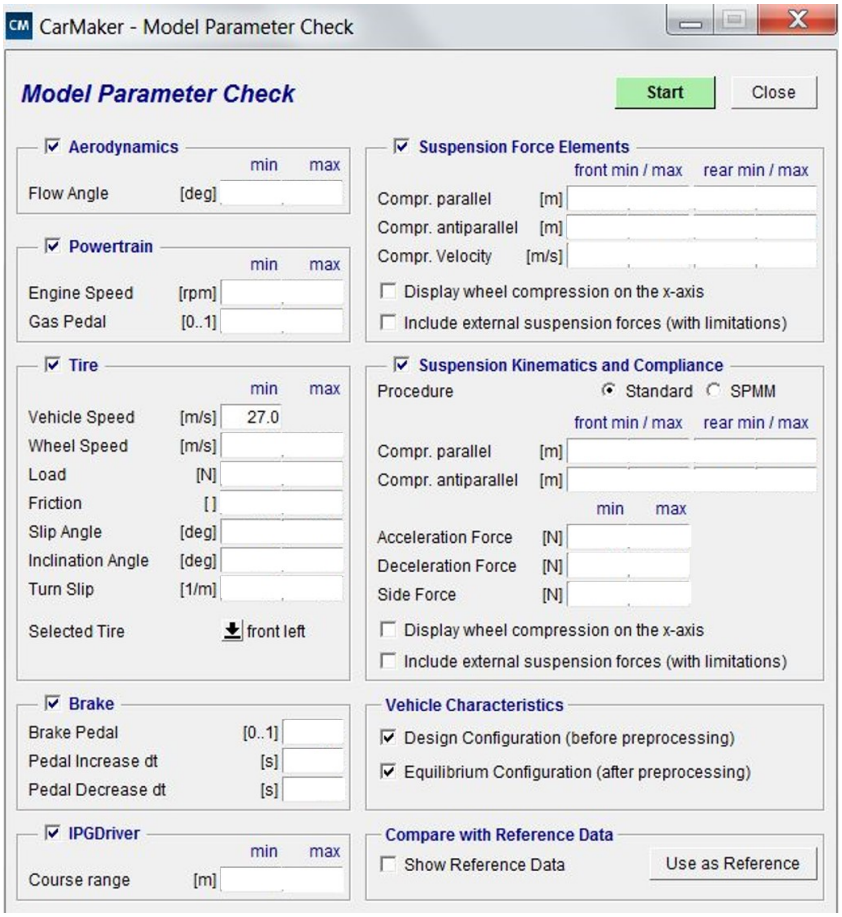


Figure 4.1: CarMaker Vehicle Setup

Additionally, the definition of the tire model used was necessary as it affected the quality of the output data obtained post simulation. Predefined values obtained from calculations with respect to the aerodynamics of the vehicle model, the weight distribution as well as the Inertia modelling of the both the sprung mass and unsprung mass of the model were plugged in to obtain results which could be validated in the future. The sprung masses values included the inertia's Ixx, Iyy and Izz for the four wheels whereas the unsprung mass values included the definition of the center of gravity position and inertia values for the chassis. The values of the above mentioned parameters could be modified in the dialog box shown in figure 4.1. [3]

The focus of this project was on the steering system. The modelling of both the mechanical and electric subsystems of the EPAS were done on Simulink. Since IPG CarMaker had an inbuilt functionality in the form of a plug-in for models in Simulink, the steering model could be both used and modified for the purpose of this project. These models were exported and initialized with the help of scripts coded in MATLAB.

4.1 Test runs

One of the primary elements which had to defined, in order to evaluate the parameters, were the maneuvers, also known as test runs within the IPG environment. The definition of these test runs required inputs from the IPG environment such as the road surface, the longitudinal velocity, the steering wheel angle and the lateral acceleration. They were defined in steps which replicate a real time test including options to define the type of driver, with either manual inputs or using the inbuilt functionality of the IPG Driver. Different steps within a particular test run involved the definition of both lateral and longitudinal dynamics of the vehicle in terms of the duration and magnitude of braking, acceleration, clutch and the steering wheel angle input. The test run could be modified and saved by adding steps between different conditions that were initially defined. [9]

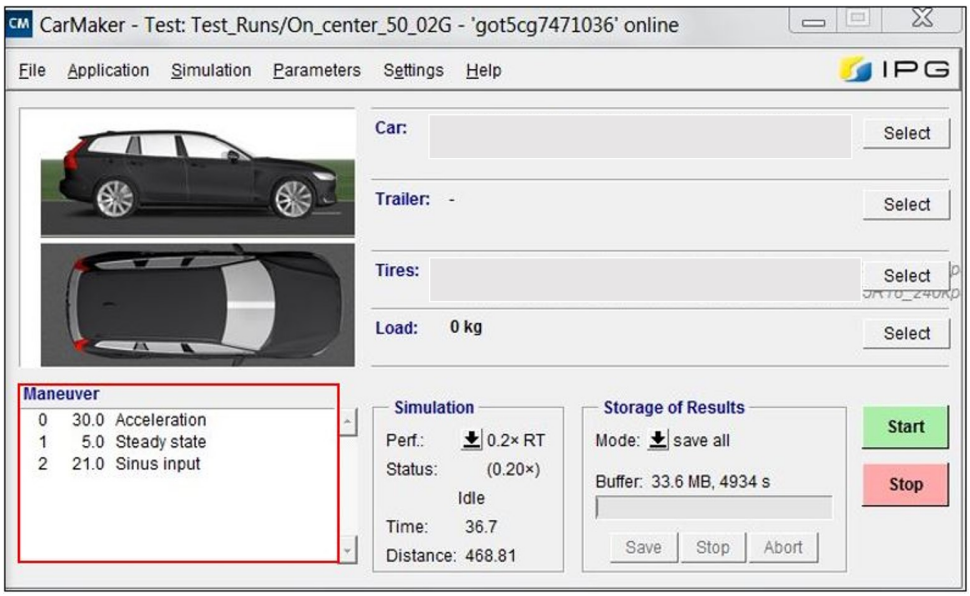


Figure 4.2: Maneuver definition

The maneuvers implemented during the course of this project were pre-defined, standardized, test procedures by Volvo Car Corporation. The definition of these manoeuvres required specific values of certain primary parameters such as the lateral acceleration, longitudinal velocity of the vehicle, the steering wheel angle magnitude, the steering wheel velocity as seen in figure 4.2. Additional options such as the definition of time period and amplitudes for sinusoidal sweeps aided the modification of these events during the simulation process.

4.2 Inputs and outputs

Once the definition of both the inputs and the maneuvers was completed the next step was to define the output quantities which would influence the calculation of the OM's chosen for the purpose of this project. Furthermore,extracting only the necessary data from these simulations,was essential in reducing the time required for post processing the data which in turn would reduce the lead time on the optimization.

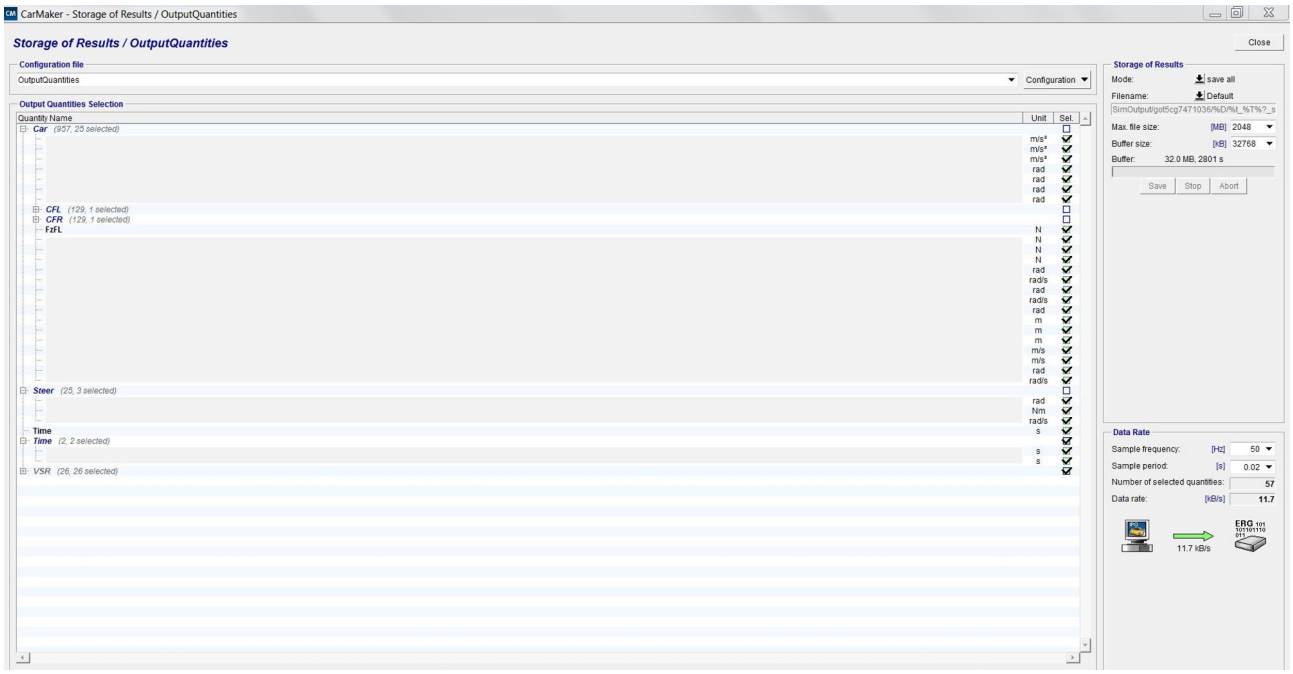


Figure 4.3: Output configuration file

A list of output quantities and saved as a separate configuration file that could be extracted and read when the data was being post processed as seen in figure 4.3. Once the configuration file was created and saved it could be pointed to a particular directory where the results files would be saved. This was also essential when the data would be post processed as the software used for processing,Sympathy for Data,required a specific folder structure. This was necessary as different sub-blocks within in the software pointed to files in different locations within this folder structure. [9]

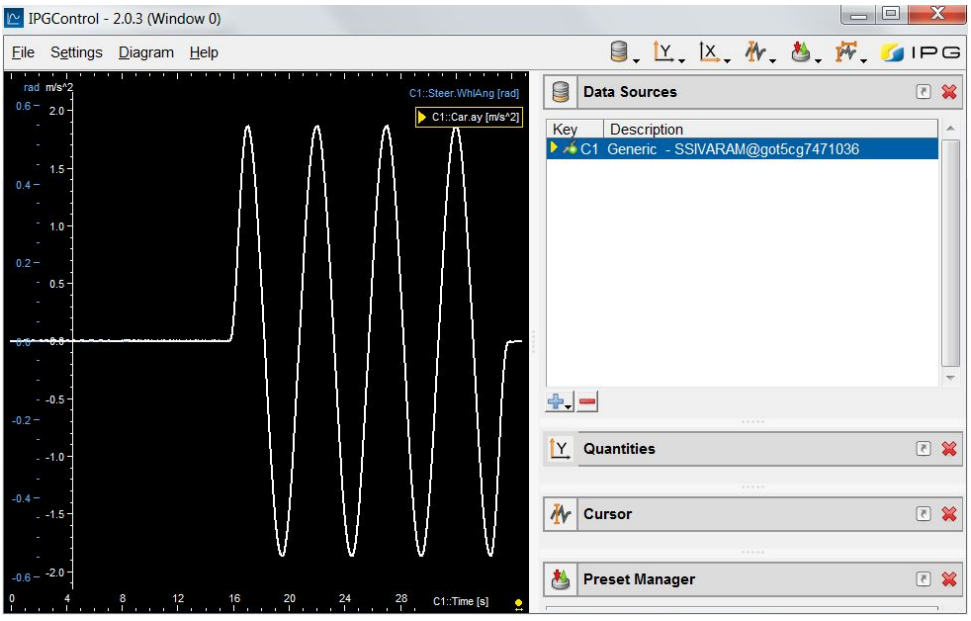


Figure 4.4: IPG Control

Another interesting yet important feature in the CarMaker environment was the ability to both visualize and plot data real-time,when the manoeuvre was being performed. This was possible with the plug-in's IPG Movie and IPG Control respectively.From figure 4.4 it could be seen that this feature enabled verification of the input data which would go into the optimization routine,at an early stage and this was essential in reducing the time required to debug the errors encountered at later stages in the project.

4.3 Interfacing

The setup of the vehicle model, the definition of maneuvers in the CarMaker environment and its ability to interface with MATLAB and Simulink have been spoken about in length. In order to perform simulations on CarMaker with different versions of the steering model, it was necessary to initially find a suitable interface which would aid in making modifications to the model at different stages.

- **CarMaker for Simulink**

  Since the Optimization routine involved running multiple iterations of the simulations in the CarMaker environment, running the Simulink model via the start-up script coded in Matlab was not possible. The reason behind this was, the start-up script included the initialization of multiple software's such as IPG CarMaker, Matlab, Simulink. Thus, the lead time to complete a single simulations on IPG CarMaker and store the result data was too long .

- **FMU**

  Hence an alternate procedure was used to run IPG CarMaker as the standalone simulation environment, with all the aforementioned data being plugged into the vehicle model remotely. This was achieved with the help of the Functional Mock-up Interface (FMI). The FMI offers the possibility to make use of models exported by third party tools in a standardized form as so-called Functional Mock-Up Units (FMU). [10]

  – **Setup**

    In order to integrate a new FMU into IPG CarMaker it had to be imported first. The FMU interface on CarMaker had multiple tabs with options to plug-in, import, configure and debug the FMU's. The FMU's were classified on the basis of the subsystem

they're modelled. The entire list of FMU's could be seen on the overview tab in the GUI as shown in figure 4.5. Based on how the FMU is configured it returned the status of either New, Ready or Error which was helpful during the debugging phase. [11]

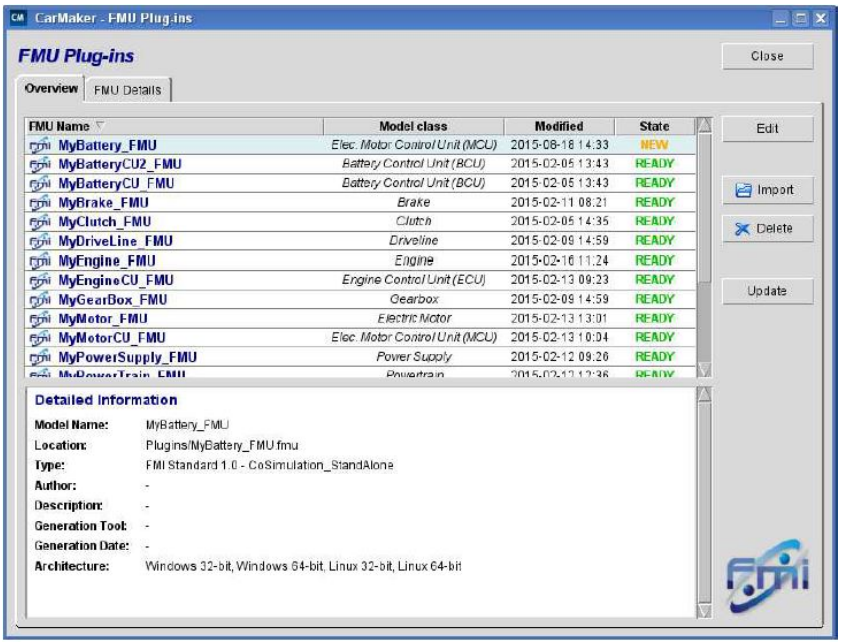


Figure 4.5: List of all FMU's [11]

– **Configuration**

To configure the plug-in, the desired FMU was chosen on the basis of the model class and then further edited.Since the focus was on the steering and the power-train subsystems, the FMU for latest version of the power-train was pre-compiled. However since this was not possible for the steering model, an FMU for a 2016 version of the steering subsystem was used instead. Once a model class had been attributed to the FMU, the FMU's inputs and outputs needed to be connected to suitable signal sources and their respective destinations in CarMaker.

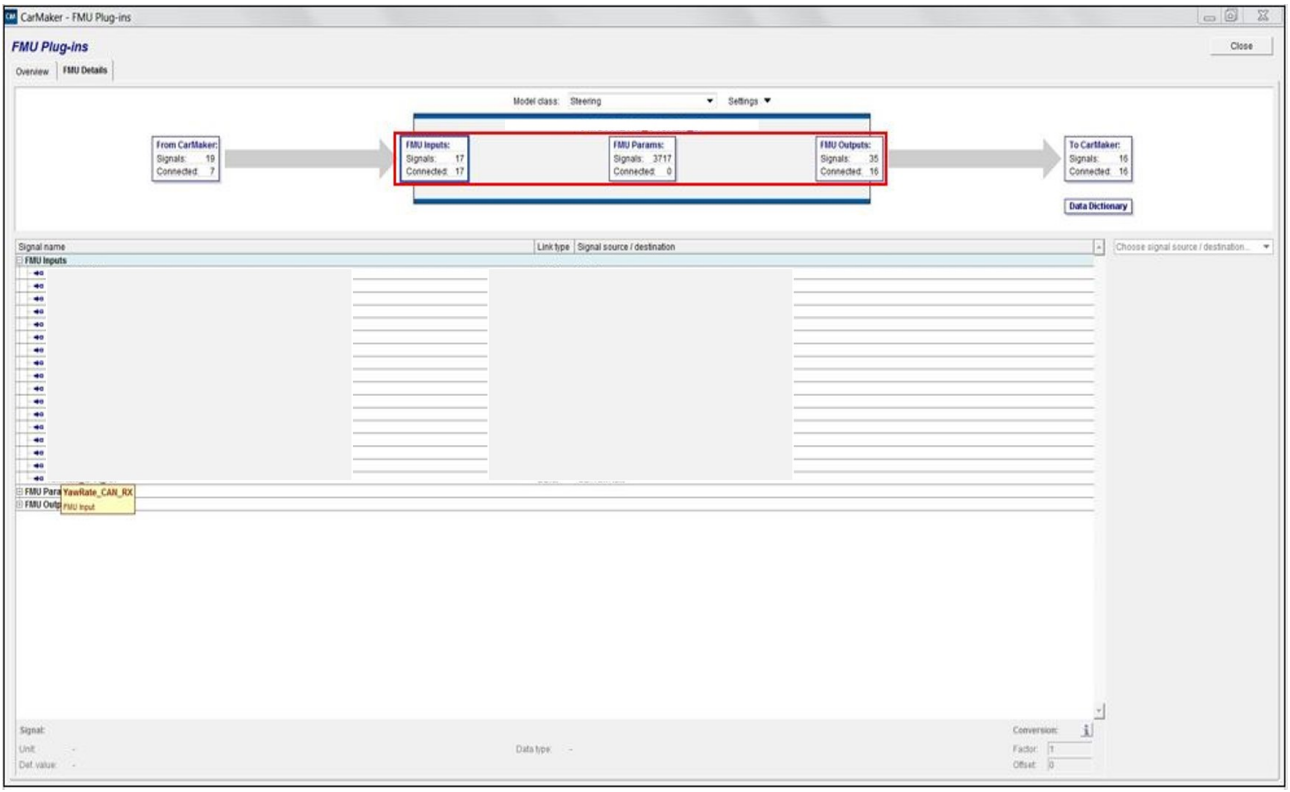


Figure 4.6: Steering FMU setup

From figure 4.6 it could be seen that there were multiple options to configure the signals. The signal sources for the FMU inputs included interface variables, datadict quantity, real time expression and constant values. Whereas the signal destinations for the FMU outputs had the interface variable and datadict quantity options. The option to add FMU inputs and outputs to the data dictionary was helpful during the debugging phase as it was possible to view if the FMU outputs were used elsewhere in the model. Once the configuration of a particular FMU was complete the GUI was closed thus saving the current configuration setting as the default setting. [12]

– **Parameterization**

The simulation on IPG CarMaker could be started via the FMU by selecting the desired FMU. Optimization of the parameters of the steering subsystem involves modification of these parameters continuously. For every FMU these parameters were stored in an internal description file in the .xml format called "modeldescription.xml". [12]

Since the parameters needed for the purpose were present in the description file for the steering model, the FMU builder was used to point to the description file containing the parameters for the particular variant of the subsystem. This description file for different variants could be parsed as well as overwritten when the input was

modified. To modify the parameters the FMU would unpack a ZIP-file which contains the description file, edit the parameters in the .xml file and pack everything again.

– **Debugging**

To avoid instances where the FMU would not behave as expected, the debugging logging was switched on. Thus keeping a track of the the internal state of the FMU during the simulation and the outputs from the FMU after the simulation. Since the project involved modifying a large number of inputs, multiple times, leaving the debug logging permanently switched on, slowed down calculations significantly and swamped the CarMaker log file with unnecessary output. This lead to reaching CarMaker's internal log file size limit, thus ignoring further logging. The size limit (about 10 MB) was chosen intentionally, and is hard-coded into CarMaker and cannot be changed by the user. [10]

* One way to handle excessive logging was to redirect the logging output from the FMU to an external file. In order to separate FMU debug logging output from regular CarMaker log messages, the following entry in the project directory's Data/Config/SimParameters file:**"FMU.Logging.ToFile = value"** was made. The field containing value could be filled in with binaries. The value 1 indicated the messages would go to an external log file called "FMU.log". The default value was 0, which indicated the messages were stored in the CarMaker log file.

* Another way to handle this issue was to suppress all FMU log messages before a given time in the simulation. This was done by adding the following entry in the project directory's Data/Config/SimParameters file:**"FMU.Logging.Start = value"**. The field value could be filled in by specifying a global time. This would indicate the when the FMU logging would start.

* Further options were available to deal with this issue where-in the C-code on the basis of which the FMU was compiled could be editted and thus helping filter the output being logged. These fixes were really helpful throughout the optimisation process as the Simulation environment (IPG CarMaker) would not crash due to excessive warnings or errors or output logs generated from the FMU.

4.4 Process automation

Following the setup of the vehicle model, test runs and the output quantities, it was evident that the process involved multiple interactions between software's. This was necessary to both modify the inputs as well as initialize the simulations in IPG CarMaker. Thus in-order to keep the optimization process efficient and robust, the next step was to automate the simulation process within IPG CarMaker. This was achieved using a combination of the Test Manager and Script Control features.

4.4.1 Test manager

A test series in the test manager basically consisted of test runs that were executed automatically one after another. The test manager also offered multiple functionalities to optimize the preparation, execution and analysis these TestRuns. This feature aided the definition of multiple variants of the same maneuver (on-center, frequency response) by modifying parameters in the Test Manager interface. All the elements within the defined test series were executed sequentially from top to bottom. [9]

- **Creating the test series**

  The optimization process involved running multiple simulations for different variants of different maneuvers. Hence, the first step in process automation within the Test Series was to create groups pertaining to different variants of the same maneuver. Figure 4.7 shows the definition of a group of on-center maneuvers with multiple variants.

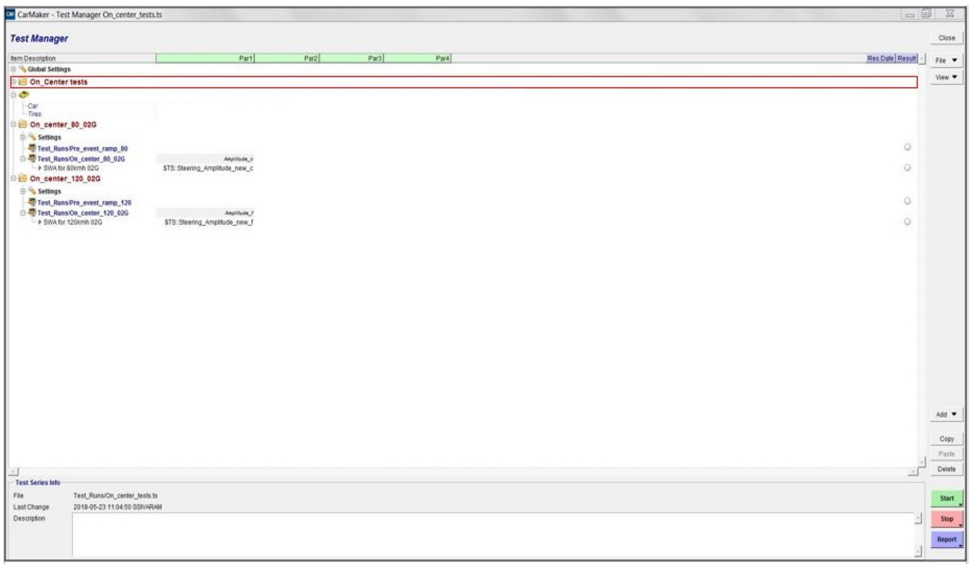


Figure 4.7: Group in Test Series

This ensured a well structured series of steps with the possibility to define different parameter settings for different groups. Once a group was created and the desired name was set, we proceeded to define the vehicle configuration. This included the definition of the vehicle name, Tyre models used, distribution of loads on the vehicle.

One of the key features which was utilized before the initialization of each maneuver was the definition of named values,key values and test space variables. These referred to variable types which could be defined by the user in the test series. These user defined variables could then be modified before the initialization of a group within the test series. [10]

The named value was thus any editable parameter defined by the user which would take effect only in a test run simulation. Within the on-center test series, the steering wheel angle amplitude of vehicle was defined as a named value using the following syntax :

[**Type -NValue Name- Amplitude Value- 20**]

The key values referred to the variables that could modify info-file keywords. This mainly applied for settings that did not have an editable parameter field in the CarMaker GUI (Eg : different tyre models, different vehicle data set).

Test space variables were auxiliary variables that are only known within the test series. They basically stored and calculate values. Within the test series shown in figure 4.7, the parameter lateral acceleration was set as a test space variable. This functionality could be created in the settings blocks of the test manager GUI with the following syntax :

[**Type -TS Name- Lateral acceleration Value- 0.2**]

- **Calculations, Criterion and Diagrams**

  An interesting feature within the Test Manager was the option to perform calculations of characteristic values both online and offline. The values obtained from these calculations, influenced the outcome of the simulation. To calculate values real-time, "real time expressions" were used. To define Real time expressions, the syntax **RTexpr** was used followed by the definition of a new quantity whose name was identical to the previously specified identifier of the characteristic value. [10]

  Real-time expressions were implemented in a few places during the setup and automation of the simulation environment :

  1. The first instance of real time expressions being used was the definition of test runs. In the maneuver dialog box of the CarMaker GUI, real time expressions were used to define the initialization and termination condition of the Test Runs.

  2. Another instance was when real time expressions served as trigger for mini-maneuver commands. Once the defined condition was satisfied, the mini-maneuver command was executed.An example of the same was **"[Car.v=0.0] Log "Maneuver finished!"**, where the following syntax was entered in the mini-maneuver commands dialog box.

  3. The final instance where real time expressions were used was in the offline calculation of parameters, via script control commands. Here the values were calculated on the basis of a user defined function after the simulation was completed by analyzing stored result data. The path to the script file containing the functions was pre-defined before the definition of the characteristic values.The scripting which was based on the TCL command language and will be discussed further in detail in the coming section.

  Features such as definition of criterion and plotting figures though available, were not utilized as the post-processing of data took place in a separate environment, independent of IPG CarMaker. Furthermore the use of these features for analysis or cross-referencing, would defeat the purpose of automating the process within the simulation environment .

- **Global and local settings**

  The settings option on the Test Series GUI gave the possibility to define configurations applicable to more than just a single Test Run.The placement of these settings, within

the Test Series was vital, as they would influence every TestRun, Variation at a lower level. The Test Manager provided two kinds of settings: Global settings and local Settings blocks.

The section Global Settings were defined at the start of the test series. The settings defined here applied to all the groups, test runs and variations present in the test series. Global Settings contained named values, key values, test space variables and the Script files. These files contained both real time expressions and script control commands coded in .tcl language.

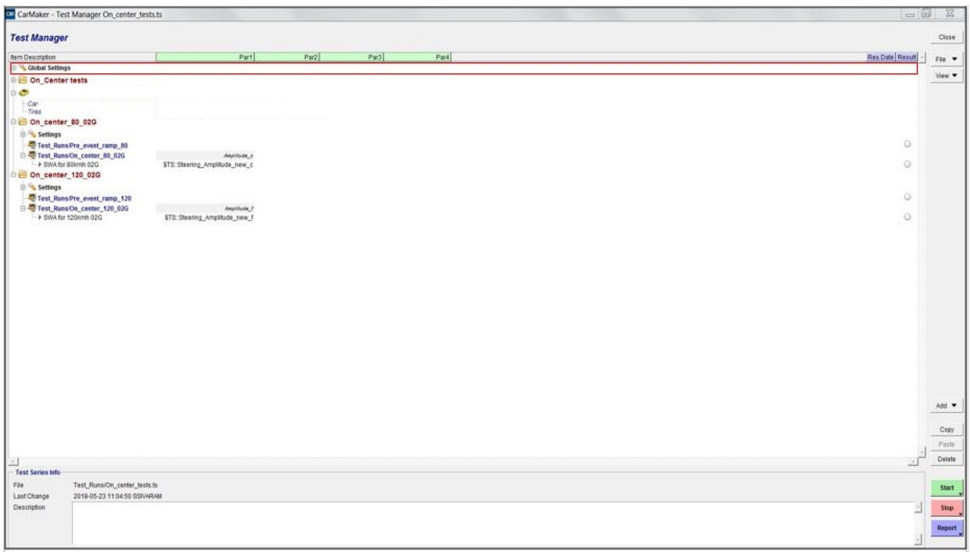


Figure 4.8: Global settings

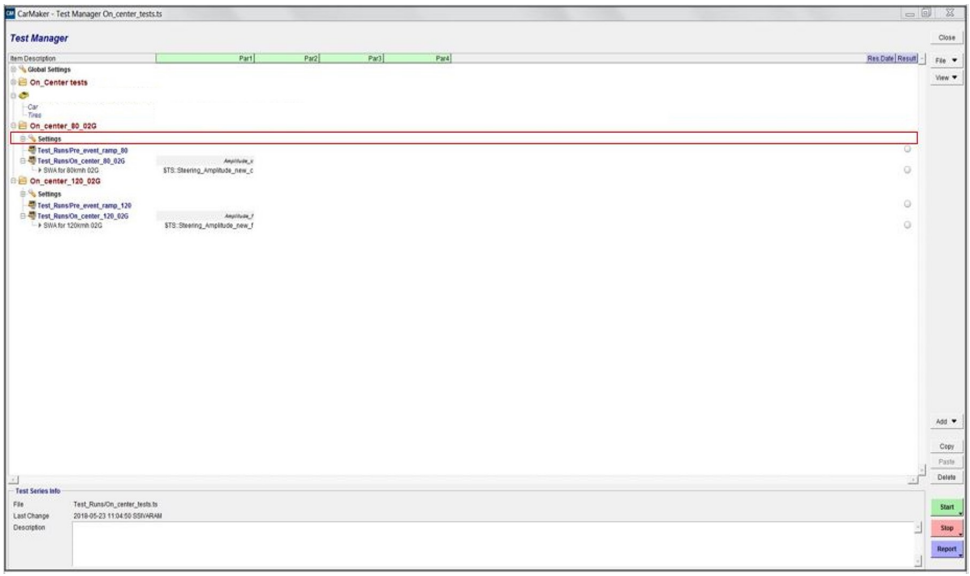


Figure 4.9: Local settings

In stark contrast the local settings block could be added at different stages of the test

series as seen in figures 4.8 and 4.9. These settings were applicable only to the level they were placed in. For Instance if the settings were placed at the top level of a group, they applied to every test run and variation within the group.

In order to define a script file within the settings item, **StartProc** and **EndProc** syntax were necessary as they were indicative of start process and end process in the .tcl language. In order to deactivate these commands a new settings blocks had to be defined with new commands which would overwrite the previously defined settings.

- **Test runs and variations**

  Following the definition of the global settings, calculations, script files the last step in setting up the test manager was the definition of the test runs and their variants.The test-series defined for the sine with dwell maneuver seen in figure 4.10 required multiple variants with each variation referring to a different value of steering wheel angle at the same value of lateral acceleration.

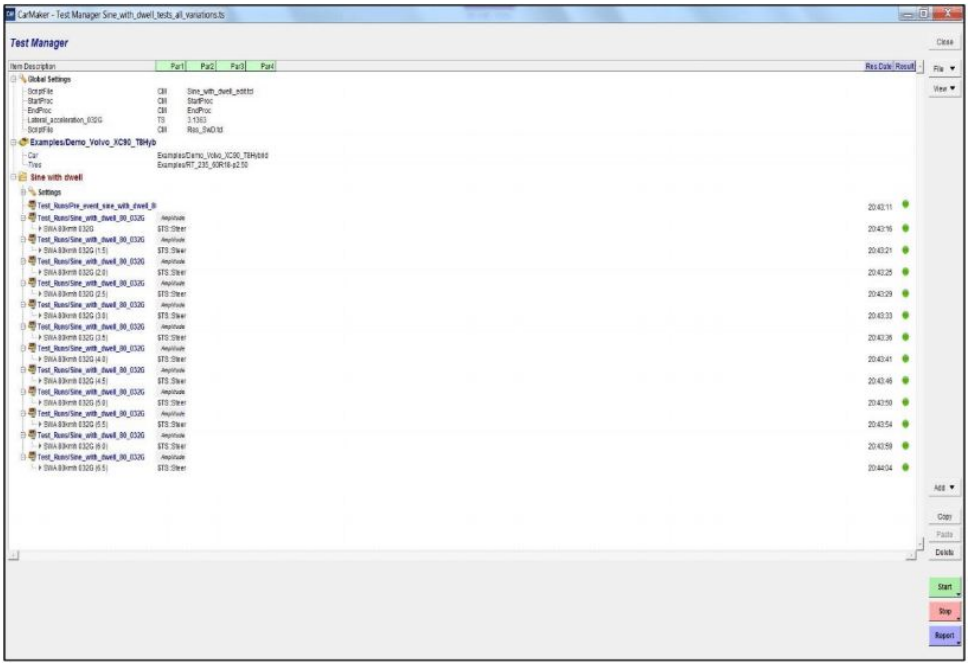


Figure 4.10: Test Runs and Variations

In order to setup the test run, the pre-defined test run which was earlier created and saved was chosen. Similarly multiple test runs could be chosen to be a part of a group within the test series. Once the test run was selected, the necessary named values, key values and test space variables in order to define the variants of the particular maneuver, were scripted. Thus an entire Test Series was setup pertaining to each of the maneuvers and their variants that were necessary to extract the Objective Metrics.

4.4.2 Script control

Script control is a tool which enabled the automation the initiation, modification and execution of various tasks within IPG CarMaker, using a simple script language (Tcl/Tk).As mentioned in previous sections variable types such as named values, key values, test space variables and script files played a vital role in the both setting up the test series and execution of the process automation. [10]

Within script control all the aforementioned variable types could be defined with specific commands. This included setting new variables within the test space, extracting the values of these variables at specified instances and the storage of results in specific directories. This was deemed necessary as the results from the simulation environment were processed independently in a different software.

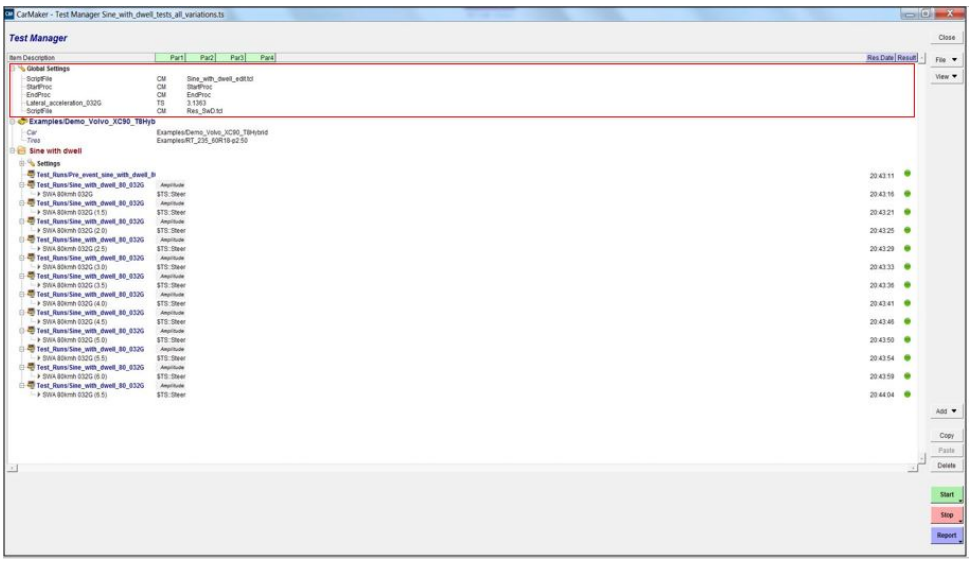


Figure 4.11: Script file in Test Manager

From figure 4.11 it could be seen that the script files were strategically placed in the Test Series, thus ensuring complete process automation of multiple maneuvers with modifications to various parameters.

- **Process flow**

  As mentioned in previous sections every test run/maneuver required a certain pre-event to extract the values of Inputs such as the Steering Wheel Angle at specific values of Lateral acceleration for different vehicle speeds. Thus every script file defined for process automation followed a specific procedure which is shown in figure 4.12.

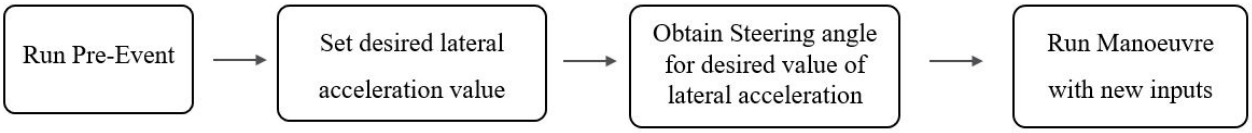


Figure 4.12: Process automation

– **Step 1**

In order to initialize the pre-event,**StartProc** command was used followed by the definition of the name of the event. This ensured CarMaker initialized the event and stored the result files from the same at the defined location. The **EndProc** was used to both terminate the pre-event and initialize the DNA maneuver, the results from which would be used to obtain the objective metrics.

– **Step 2**

Once the pre-event was terminated , the next step was the preparation of the results file from the pre-event and the definition of the test space variables within the script file. This syntax ensuring all the named Values in the settings block would be prepared for the desired value of the test space variable. The test space variable in this case was a particular value lateral acceleration of 0.2 G's. The real time expression ensured that CarMaker went through all the values for steering wheel angle for values of Lateral acceleration lesser than or equal to the specified value.

– **Step 3**

Following this, the corresponding value of steering wheel angle at the specified value of lateral acceleration was found. The next step was to replace the named value **Amplitude** with the value obtained. The code primarily ensured the previously defined value for amplitude would be erased. Following which a new test space variable was created.The value of the new test space variable was obtained from the result files of the pre-event as mentioned in the previous step.

– **Step 4**

The script file also included the possibility to generate variations using specific commands. Following the completion of simulations, the result files were stored in a pre-defined directory, accessible to the post-processing environment "Sympathy for Data".

5. Optimization Environment

5.1 Esteco ModeFrontier

With the simulation environment set up, the next step was to automate the entire procedure for optimisation. This was carried out using the optimisation software, ModeFrontier by Esteco. The vehicle maneuvers were iterated in ModeFrontier, and the outputs then optimised. The flow for this optimisation study is described in further detail in section 5.2.

Though ModeFrontier is a very powerful tool capable of syncing with most software available in the market, it does not have direct link nodes for either IPG CarMaker or Sympathy for Data. Hence these links needed to be coded, for which a mixture of Python, TCL/TCP, and MS DOS functions were utilised. Also, for an effective optimisation routine, it is required that there should be no user inputs, with a fully automated process. Additional scripting was carried out in order to make this possible and will be discussed along with the others in the following sections as well. Another goal for the project was to make the tool as user-friendly as possible, so it is not a burden for the test engineers to use.

With multiple software interfacing with each other, and various test maneuvers taking place, each simulation was expected to take a considerable amount of time. And with around 500 to 1000 simulations per optimisation run, this time needed to be kept within check in order to have feasible optimisation runs. This too has been further discussed in the following sections.

5.2 Process flow

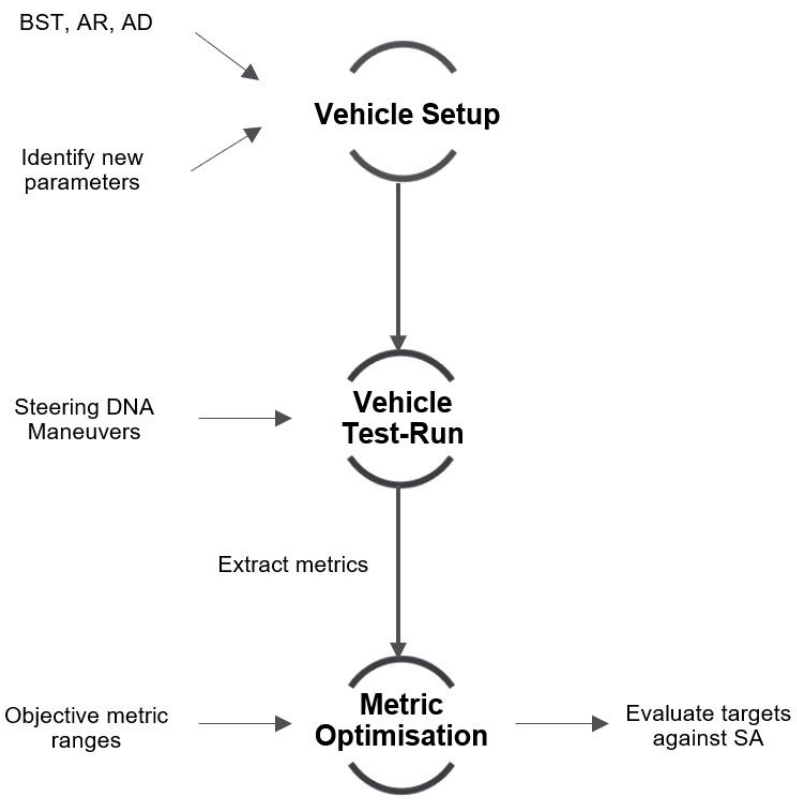


Figure 5.1: Optimisation Process Flow

The process flow, as shown in figure 5.1, can be broken down into three major components. Each of the components will be studied in further detail in the coming sections.

1. Setting up the vehicle for a test.
2. Performing the test with the required maneuvers.
3. Extracting the test metrics and optimisation.

The ModeFrontier Flow was modelled to mimic the aforementioned process flow, with three sections of the flow and blocks in each section working to achieve the desired responses. This flow is seen in figure 5.2.

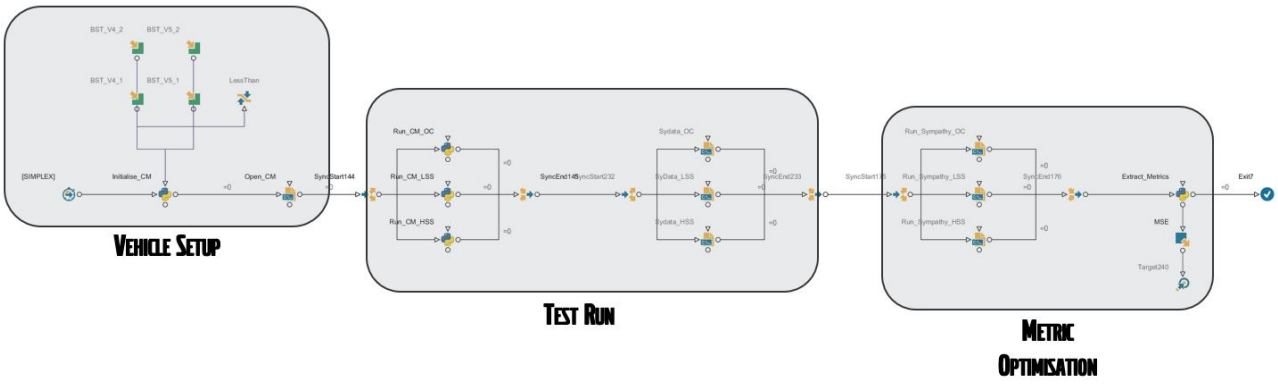


Figure 5.2: ModeFrontier Process Flow

5.3 Vehicle setup

The first part is setting up the vehicle. The physical components of the vehicle will remain the same throughout the series of tests. The parameters altered are the ECU parameters of the steering, and as mentioned earlier, the steering boost curves, the active return, and the active damping will be dealt with first. The study was undertaken for these three parameters, but the work shown and results discussed henceforth, will specifically be with respect to the steering boost curves. The routine can be later mimicked for both the active return and active damping functions.

In case of the boost curves, there consist 10 input parameters.

- 6 boost curves, each corresponding to a specific speed
- The different vehicle speeds at which the boost curves are defined
- Maximum rack force at each speed
- Maximum rack force possible
- Maximum assistance torque

The parameters other than the 6 boost curves are pre-determined through physical limitations or calculations, hence it was only the 6 curves which needed to be optimised. However, the boost curves are vectors consisting of 9 values each, with every value describing a point on the curve. So the optimisation needed to account for the curve and its properties.

The initial approach to this problem was to move each point on the curve, optimising it to its best value. The curve connecting all the optimised points would be the optimised curve. However, there were two major problems with this approach.

1. With a total of 54 inputs, the optimisation run would not be effective unless it is evaluated for a very large number of iterations.
2. If each point is optimised individually, there is a possibility that the final curve will not be smooth, and this is not a comfortable setting for the driver.

Hence, instead of looking at the curve as a collection of points, it was studied as a single entity. A function would be constructed to fit all the given points of the curve, and then the coefficients of the function would be optimised to get the best curve.

The Curve Fitting Toolbox on Matlab was first used to try and generate the function, which could then be used in optimisation. The Curve Fitting Toolbox generated a second-degree exponential equation 5.1.

$$f(x) = 1.987 * e^{-0.02434.x} - 0.5177 * e^{-2.371.x} \tag{5.1}$$

This function was built into the optimisation problem and a few tests were conducted. However, it was seen that the function was still too specific and was unable to explore the entire design space. Minor changes to the coefficients of the function brought about large changes in the curve and it was not possible to use continuous values to explore the whole space.

The function needed to be simplified, display the trend of the curve and explore the extent of the design space. From the spread of the points in figure 5.3, it was deduced that a logarithmic curve as in equation 5.2 would be used. However, a few added constraints were required due to the nature of the boost curves, specifically,

1. The curve would need to start from the origin.
2. A boost curve for a higher speed requirement needs to have points of higher values than that for a lower speed requirement
3. The steep rise of a general logarithmic curve needs to be reduced.

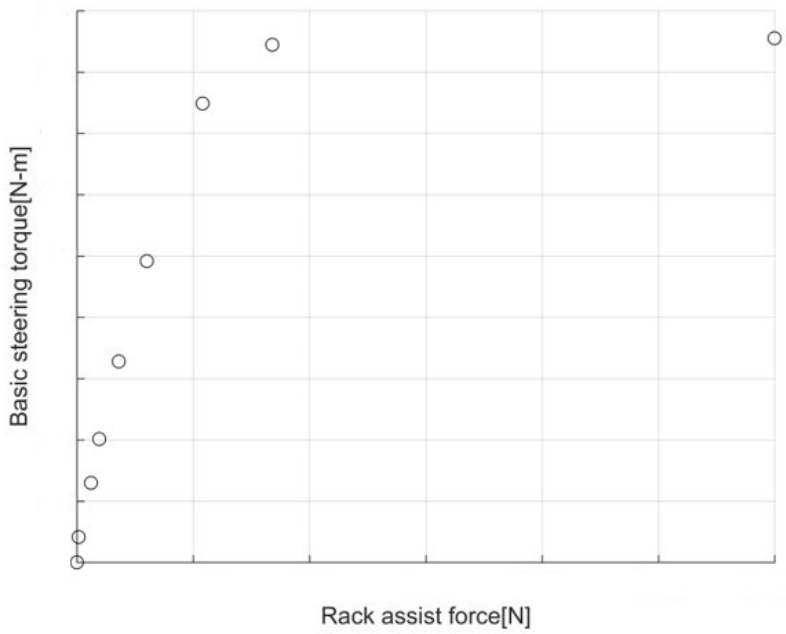


Figure 5.3: Boost Curves - Initial Values

$$f(x) = log_A(x) \tag{5.2}$$

With these constraints, the general equation of the boost curve finally used was equation 5.3.

$$f(x) = A * ln(x + e^{\frac{B}{A}}) - B \tag{5.3}$$

A major advantage of this design is that the equation further reduces the number of input variables to 2 per curve, making it a total of 12 input variables.

In order to verify the curve, to see if it able to explore the entire design space, a Matlab slider function was created and the curve was checked at various values of A and B.

With the optimisation concept finalized, this was implemented into the ModeFrontier flow. To do so, 6 vector input variables were created in ModeFrontier. Each of these inputs contained two variables, defining the values of A and B for the corresponding boost curve. These variables are altered by the software at every iteration of the optimisation. The upper and lower bounds of these values were determined from the physical limitations of the rack force and steering torque. The bounds are as follows,

A = {0.800, 1.000}
B = {0.000, 6.000}

The boost curve data is stored as an array of points in a .xml file, which has a parent-child structure as seen in figure 5.4. However, the inputs provided for each simulation from the optimisation environment are the coefficients of the equation which fit the curve formed by these points. Hence there is the need for a process to calculate the point data from the given coefficients of the curve. To do this, a python script is used. The script gathers the following data in order to compute the array of points for the new simulation.

1. obtains the values of the equation coefficients from ModeFrontier,
2. parses the .xml file to obtain the x-values of the boost curve,
3. obtains the equation type from the user at the start of the optimisation.

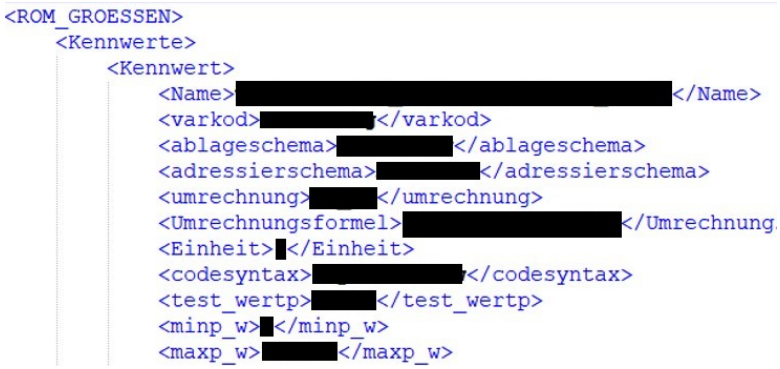


Figure 5.4: .xml File Structure

Once the equation is solved, the script uses a find-replace function, parses the XML and replaces the old configuration with the new one.

5.4 Test runs

The test runs for the various SDNA maneuvers were initially performed in CarMaker for Simulink, since the Software-in-Loop steering module was modelled on it. Since CarMaker was to be accessed in Simulink, the easiest way to do so through ModeFrontier was to use a Matlab script. The cmguicmd commands were used in order to do so. The code for the same is given below.

```
CarMaker_VSR_Env
sim('CarMaker_VSR_Env')
cmgiucmd('LoadTestRun "C/Thesis2018/..." ')
cmguicmd('StartSim', 0)
cmguicmd('StopSim', 60000)
```

This was a simple solution, however, it came with a few problems. The main problem was the slow Simulink interface, which required nearly a minute or two to open the CarMaker environment. This cost valuable time during the runs, increasing the overall optimisation time by nearly 12 to 16 hours. Another big disadvantage was licensing. With Matlab having limited licenses, and with a single computer opening and closing nearly 500 iterations of Matlab in a day, there were problems at the servers.

Hence this interface needed to be altered, and Python was used instead of Matlab as the connection between CarMaker and ModeFrontier. The maneuvers were coded in TCL, which was referred using the Python scripts. And since Python was being used, the steering Simulink model needed to be replaced with a functional mock-up unit (FMU) which is explained in detail in the coming section. This process greatly reduced the interaction time between the software, facilitating faster optimisation runs. In addition to this, multiple instances of CarMaker could be run on different ports, with Python providing different TCL scripts to each port, allowing the optimisation environment to run the simulations in parallel. To run CarMaker on different ports, batch scripts were utilised.

A functional mock-up unit, or FMU, is a tool independent standard to support both model exchange and co-simulation of dynamic models using a combination of XML-files and compiled C-code. [11] The Simulink steering model was converted into an FMU, which was then imported into the Functional Mock-Up Interface of CarMaker, which allows the FMU to be dynamically linked and co-simulated in the CarMaker environment.

5.5 Metric extraction

With the vehicle test runs carried out, the data was post-processed using Sympathy for Data. Sympathy was executed using batch scripts, first running the .sydata flow to configure the CarMaker results, followed by the maneuver flow.

Sympathy for Data is made to be used by engineers who want to look at the characteristics

and performance of the vehicles that were tested or simulated. Hence, it provides documents in MS Excel and PDFs with graphs and other visual aids to help understand the data. However, since Sympathy is being used as an element in the optimisation process to extract the values of the metrics and feed the data back for optimisation, the processes for the generation of these documents were terminated and instead csv files were generated with metric data. A python script was then used to extract the data from the csv files and fed back into the ModeFrontier environment. This was one of the major time savers in the project, cutting down the time of a single run of Sympathy from about 4 minutes, to under a minute.

6. Metric Optimisation

6.1 Single-objective optimisation

Single objective optimisation, as the name states, is the optimisation of a single objective function. Though it seems counter-intuitive to use single objective optimisation for the problem in hand, we do so to simplify the optimisation process, thus improving the chances of reaching an optimum value and to do it quickly. The multi-objective problem is converted into a single-objective optimisation problem by utilizing a cost function. This cost function, which is an indicator of how much the current model varies from the ideal model, is defined using normalised values of the outputs and the targets, obtained from the Steering DNA Sheet. The final cost function used is given in equation 6.1.

$$costfunction = \frac{1}{n}\sum_{i=1}^{n} abs(t_i - y_i) \tag{6.1}$$

where n = Number of parameters,

t_n = Normalised Target,

y_n = Normalised Output

The final objective of the optimisation is to minimise this cost function, to attain a configuration as close to that of the ideal vehicle. Also, since the values were normalised to between 0 and 1, the absolute value of the difference was used instead of the squared value, which would have subdued the large errors instead of amplifying them.

6.2 Design of experiments and sensitivity analysis

With a large number of input parameters in the optimisation problem, the runs were not effective unless a large number of iterations were run. In order to increase the effectiveness of

each optimisation run, and reduce the number of redundant parameters, a sensitivity analysis was planned. A sensitivity analysis is used to detect the most influential parameters in the optimisation by studying both the individual effects that the parameter has to the results, as well as the interaction effects between multiple parameters.

In order to obtain a sensitivity analysis, a Design of Experiments was set up. The Uniform Latin Hypercube Method was used to generate the DOE table and the sensitivity analysis was then carried out in ModeFrontier. On studying the results of the sensitivity analysis, which can be seen in figure 6.1, it is observed that the basic steering torque curves V4 and V5 are the ones which have the highest influence on the results. Hence the consequent optimisation runs are carried out using these parameters only.

Variable Name	Output1	Output2	Output3	Output4	Output5	Output6
BST_V0_1	9.48E-04	0.00765722	9.42E-04	0.012064835	3.04E-04	0.014755382
BST_V0_2	0.008206157	0.005496685	0.009604337	0.00388923	0.00874414	0.020431272
BST_V1_1	5.82E-05	0.010374249	4.42E-04	3.43E-04	9.31E-04	0.022207059
BST_V1_2	0.011312215	0.005278745	8.48E-05	0.003764422	0.001012214	0.008316673
BST_V2_1	0.007947863	7.47E-04	0.004137661	0.003036721	0.007167388	0.062275341
BST_V2_2	0.014779157	0.014900188	0.001157677	0.001114602	0.0028903	3.52E-04
BST_V3_1	0.049089555	0.004740292	0.001637398	0.007761291	0.002275025	0.003538339
BST_V3_2	0.022124399	0.00998665	0.007448598	0.002628156	0.003530387	0.001206519
BST_V4_1	0.238071568	0.041631004	0.032850906	0.120611505	0.060210915	0.160582643
BST_V4_2	0.622982662	0.227854992	0.065463366	0.173765192	0.09166455	0.468803338
BST_V5_1	4.95E-04	0.580133428	0.323125879	0.323521271	0.330634166	0.123982278
BST_V5_2	0.003433889	0.070235933	0.541710899	0.311139811	0.478761984	0.004645844
BST_V6_1	0.009103532	0.017960004	0.010596885	0.03181814	0.01091118	0.013278072
BST_V6_2	0.011447896	0.003003204	7.98E-04	0.004542044	9.62E-04	0.095625086

Figure 6.1: Sensitivity Analysis

7. Results

The in-built post-processing capabilities of ModeFrontier are utilised to visualise the data and draw conclusions. Data for both Simplex and MOGA are plotted and studied. First, a history plot for the error in the cost function is studied as can be seen in figure 7.1 and figure 7.2. The minimum error in the Simplex run was found to be 0.682 and the minimum error in the MOGA-II run was found to be 0.779.

Since the optimization routine was being studied and validated, they were performed on an older version of the steering system. By doing so, there was access to a vehicle tuned version of the steering system which could be the basis to compare the results with. In the coming figures, the value obtained by the optimization routine is denoted by the red curve and the pre-tuned value is denoted by the blue curve. The red curve, which shows the optimal value as obtained by ModedFrontier, initially starts as a flat line with y=0, and slowly builds up as the optimization progresses.

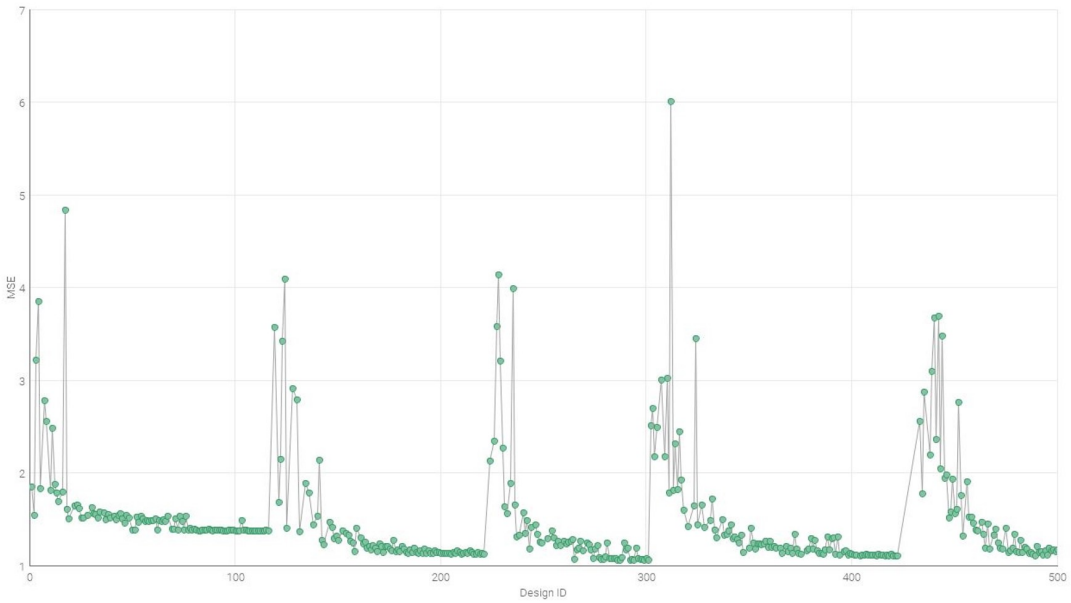


Figure 7.1: History Plot - Cost Function (Simplex)

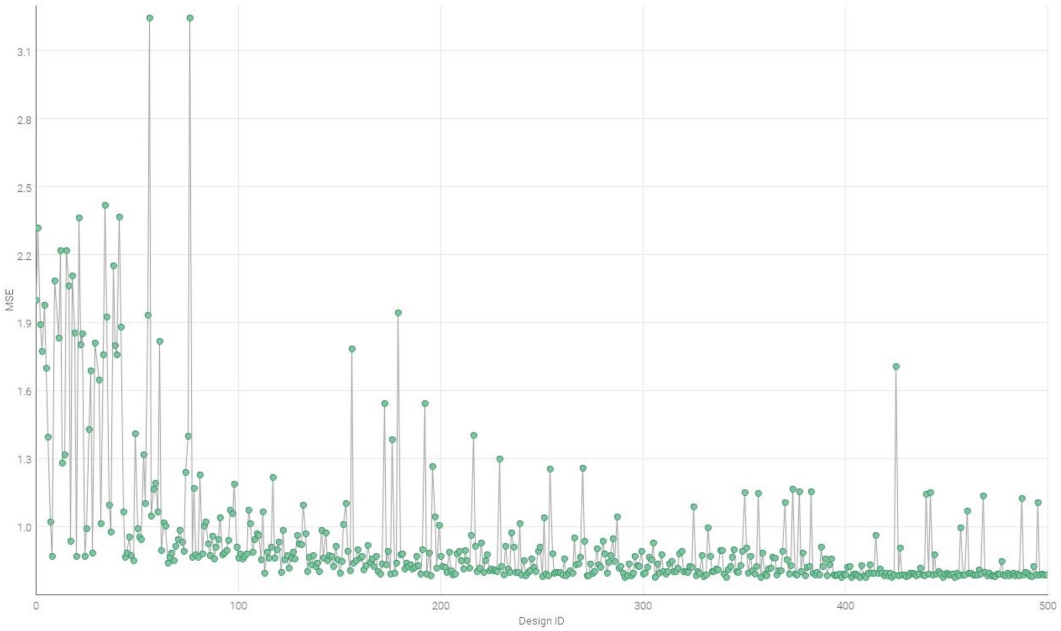
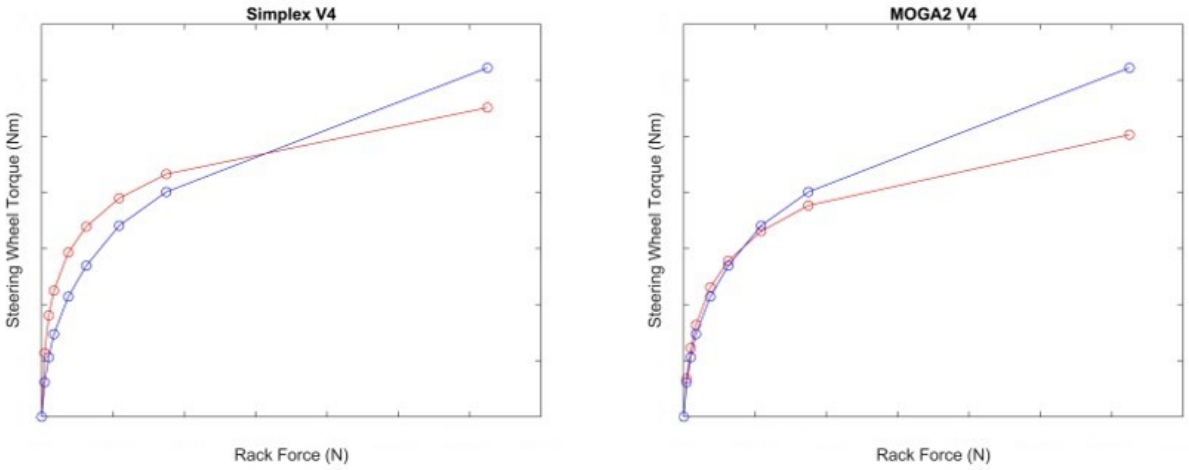


Figure 7.2: History Plot - Cost Function (MOGA-II)

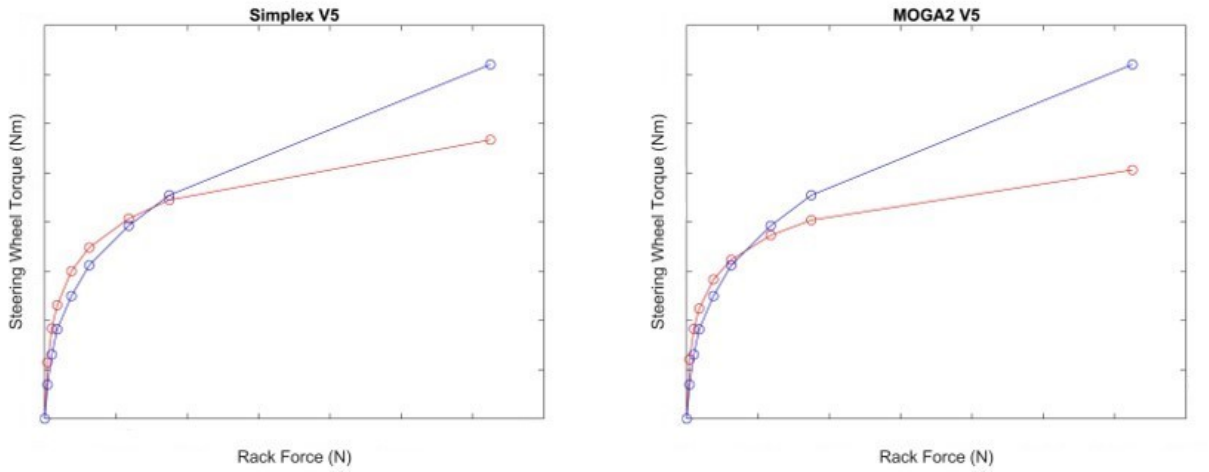


(a) Simplex

(b) MOGA-II

Figure 7.3: Optimised Basic Steering Torque V4

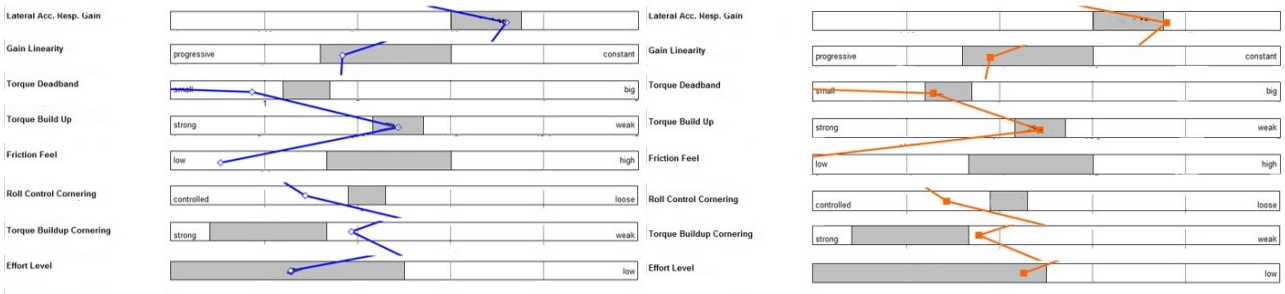
The optimisation was first run at country road speeds (V4), using both algorithms, as can be seen in figure 7.3. It was then run at highway speeds (V5), as can be seen in figure 7.4.



(a) Simplex (b) MOGA-II

Figure 7.4: Optimised Basic Steering Torque V5

The results are finally plotted onto the steering DNA sheet as shown in figure 7.5, and the 8 steering metrics are studied.



(a) Simplex (b) MOGA-II

Figure 7.5: Steering DNA

8. Conclusion

8.1 Main Findings

In both the optimisation runs, as seen in the history plots in figures 7.1 and 7.2, we can see spikes in the error, where the optimiser has attempted to mutate the data. Mutation is required to avoid being stagnant at a local optimum, however is not done too often as that causes the optimiser to randomly guess which can cause it to run for longer periods of time.

One observation, in particular, is that both algorithms achieve their design with minimum error within the first 300 iterations, making the next 200 iterations inconsequential. With one optimisation run lasting between 36 to 48 hours, there is a lot of potential with respect to time-saving if the optimisation can be terminated more efficiently. With both optimisation algorithms producing similar values of the cost function, there is not much to choose between them in this respect.

Moving onto the basic steering torque curves, in the figure 7.3a, which is the Simplex optimization of the basic steering torque, the curve is observed to slightly overestimate the steering torque in the on-centre region and slightly underestimate it in the off-centre region. However, it does follow it relatively close and should provide a similar driving feel as the pre-tuned value.

In the figure 7.3b, which is the Genetic Algorithm optimization for the same, both the optimised and pre-tuned curves are near identical in the on-centre region. However, the MOGA optimization grossly underestimates the off-centre portion of the basic steering torque. This would lead to good inputs with smooth driving but would not provide enough torque feedback to the driver in cases of sudden or jerky inputs.

Similar trends are seen in figure 7.4, which is the optimised basic steering torque curve when the car is travelling at highway speeds. With the Simplex optimisation, shown in figure 7.4a, the on-centre performance is better, with the curve not overestimating the steering torque by a large margin. Though, the off-centre performance is relatively poor as compared to the lower speed optimisation. This trend is observed in the Genetic Algorithm optimisation too, as seen

in figure 7.4b. The off-centre performance is poor and the final point in the curve is being underestimated.

On observing the generated DNA sheet, it is found to be very reasonable, with a major deviation primarily in the values of friction feel, and minor deviations in the cornering metrics.

The poor off-centre performance was something to consider, and a few reasons for this are discussed.

- The number of off-centre metrics used in the optimisation are much lower than the number of on-centre metrics. This could be countered by appropriately weighing the off-centre metric, giving it higher influence during the optimisation routine, causing it to better optimise the points.
- Achieving the peak force values during most tests is very difficult and in some cases nearly impossible with the given tests. The one test which comes close to achieving the maximum value is the Parking Effort Test. However, this test was not performed during the optimisation routine due to certain aforementioned problems, and hence that portion of the curve was not properly optimised. The maximum defined rack force is 12500 N, however, the maximum achieved by the logger during the tests performed was just shy of 8500 N.
- The biggest problem with the optimisation was the final point in the basic steering torque curves, which denote the maximum rack force. However, the test engineers at the track also do not optimise this final point and instead use the predefined value sent by the steering ECU manufacturer. The optimisation routine could be tuned in a similar way, to keep this point constant while optimising for the rest. This would improve the off-centre performance considerably.

8.2 Subjective Evaluation

A test driver at Volvo helped out with testing the optimised function on track and providing feedback on the behaviour of the vehicle, and the comments given were recorded and summarised.

The curves obtained from optimisation using the Simplex algorithm were used for the testing. Since the on-centre behaviour was adequate using both methods, the decision was taken with

respect to the off-centre characteristics. With a high off-centre deviation from the pre-tuned car, the MOGA optimised curves would result in minimal feedback from the steering at high speeds when a significantly rapid input is provided. This could make it unsafe for the driver and hence the Simplex values were used.

During testing on the High Speed Oval, the primary comment from the test driver was that the steering felt good at low speeds, but light at high speeds. This was an expected outcome since the V4 BST curve (at 80 kmph) gave a lower error with the pre-tuned curve, as compared to the V5 BST curve (at 120 kmph). Another recurring comment, which was primarily felt at the High Speed Handling Track, was that the steering felt light off-centre but had good on-centre characteristics. Again, this was an expected result, with the reasons for why the optimisation was unable to predict off-centre characteristics, discussed previously.

The final conclusion from the test was that the optimisation routine was a very good and quick method to achieve the baseline values for a pre-series car, but final manual tuning and testing would be required before the car is ready for production.

Bibliography

[1] G. Gil Gómez, "Towards efficient vehicle dynamics development : From subjective assessments to objective metrics, from physical to virtual testing," Ph.D. dissertation, KTH, Vehicle Dynamics, 2017.

[2] R. Z. Hao Chen, Yali Yang, "Study on electric power steering system based on adams," in *Procedia Engineering 15 (2011) 474–478.* Elsevier Ltd., 02 2016.

[3] M. Ljungberg, M. Nybacka, G. G. Gómez, and D. Katzourakis, "Electric power assist steering system parameterization and optimisation employing computer-aided engineering," in *SAE Technical Paper.* SAE International, 04 2015.

[4] V. Ciarla, V. Cahouet, C. C. de Wit, and F. Quaine, "Genesis of booster curves in electric power assistance steering systems," in *15th IEEE Intelligent Transportation Systems Conference (ITSC 2012).* HALarchives-ouvertes.fr, 09 2012.

[5] R. Rituraj, "Multiobjective Optimization Package of Modefrontier: The State of the Art Survey," 2017.

[6] V. Eglajs and P. Audze, "New approach to the design of multifactor experiments," *Problems of Dynamics and Strengths.*

[7] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, 1965.

[8] G. Mosetti and C. Poloni, "Aerodynamic shape optimization by means of a genetic algorithm," *5th International Symposium on Computational Fluid Dynamics*, 1993.

[9] IPG, "Carmaker user's guide version 6.0.4," 2017.

[10] ——, "Carmaker programmer's guide version 6.0.4," 2017.

[11] M. Association. Functional mock-up interface. [Online]. Available: https://fmi-standard.org/

[12] Dassault, "Fmi kit for simulink version 2.4.0," 2017.

TRITA -SCI-GRU 2019:016
ISSN 1651-7660