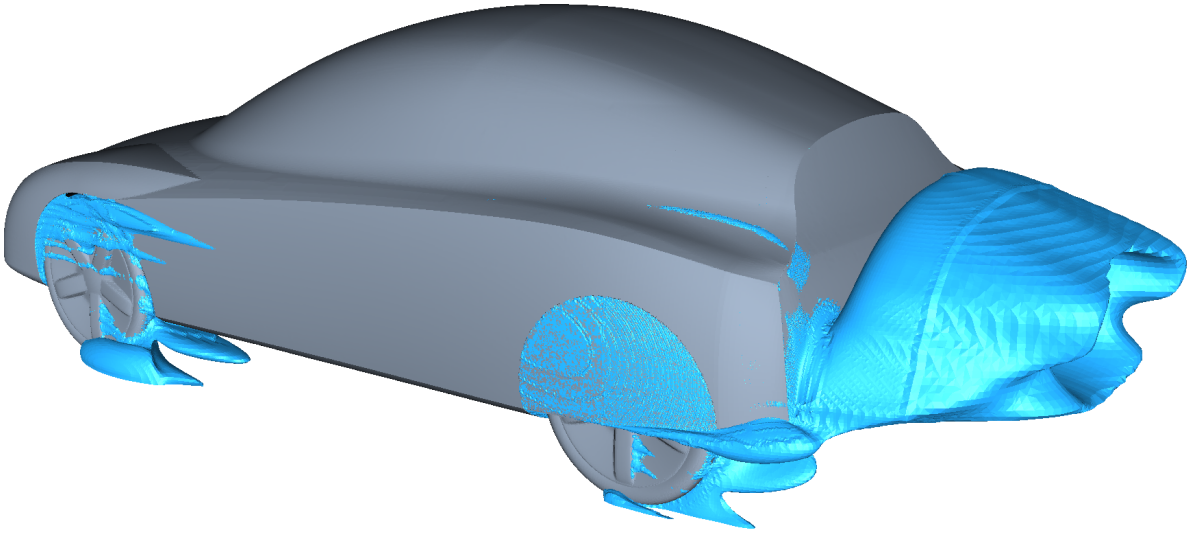
CHALMERS



Efficient Automatic Vehicle Shape Determination

using Neural Networks and Evolutionary Optimization

Master's Thesis in Applied Mechanics

ANTON LUNDBERG

Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2014

MASTER'S THESIS 2014:18

Efficient Automatic Vehicle Shape Determination

using Neural Networks and Evolutionary Optimization

ANTON LUNDBERG

Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2014

Efficient Automatic Vehicle Shape Determination
using Neural Networks and Evolutionary Optimization
ANTON LUNDBERG

Master's Thesis 2014:18
ISSN 1652-8557
Department of Applied Mechanics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 (0)31-772 1000

Cover:
Isosurfaces where total pressure is zero for optimized ACC model.

Chalmers Reproservice
Gothenburg, Sweden 2014

Abstract

Recent years have seen decreasing emission limits for passenger cars put in place to battle climate change. There is a need for car manufacturers to apply state-of-the-art techniques in order to be able to further reduce emissions and meet these new limits. Improving the aerodynamic shape of a vehicle still holds a large potential for cuts in emissions.

A fast method for vehicle shape optimization have been developed using recent years' advancements in neural networks and evolutionary optimization. It requires the construction of morphing boxes as the only manual work, with everything else being automated. The proposed method enables a study of several design parameters to be carried out in a short period of time. This is great improvement over a classical approach of changing one parameter at a time.

The optimization method is a type of two-level optimization. This means that the optimization is performed on a solver approximation instead of the real solver. This considerably reduces computation time. First a database is generated from simulations on a number of vehicle shape configurations. The configurations are chosen using a latin hypercube sampling where the minimum distance between any two points is maximized. The database is used to train a neural network to act as an approximation to the simulations. Finally an optimal vehicle shape is determined from the neural network using particle swarm optimization. The method can handle multiple objectives.

The method was incorporated in an optimization tool compatible with Volvo Car Group's CAE process. The optimization tool was used on a simplified low-drag car model in a study of realistic changes of five design parameters. An improved shape with a 12.6% lower drag coefficient (C_D) was achieved. The prediction error of C_D was 0.3%.

Acknowledgements

This master's thesis is part of a larger project in collaboration between ÅF Industry AB (ÅF), Volvo Car Group (Volvo) and Energimyndigheten (through FFI, Fordonsstrategisk Forskning och Innovation). Advisors for the thesis are Vijay Shankar (ÅF) and Alexander Broniewicz (Volvo). Examiner is Lars Davidson (Chalmers).

I would like to thank everyone involved in this project, especially Per Hamlin for his kind help and Vijay Shankar for his helpful advice.

Contents

1 Introduction

The concentration of greenhouse gases in the atmosphere is the highest it has been in the last 800,000 years (IPCC, 2013). This has already increased the global temperature and the future effects are potentially very harmful. To battle climate change the European Union issued a climate policy called the "20-20-20" targets in 2009. The main objective of this policy is to reduce the emissions of greenhouse gases by 20% in 2020 compared to 1990 levels (European Commission, 2014b).

Data from 2010 shows that the transport sector accounts for 31.7% of the entire energy consumption in the European Union (European Environment Agency, 2012). Part of the EU policy is a limit of CO_2 emissions for passenger cars. The limit for 2020 is set to 95 g/km with heavy penalties for excess emissions (European Commission, 2014a).

Improving the aerodynamic shape of a vehicle is an important factor in reaching the above limits for car manufacturers. At highway speed aerodynamic drag makes up more than 50% of the total driving resistance (Barnard, 2009, p.54). Wood (2004) estimates that the energy used to overcome aerodynamic drag makes up 25% of the total energy consumed in the United States. Hence, reducing the drag can make significant improvements in fuel economy and emissions.

Computer aided engineering (CAE) is today a vital part of car design. Normally the drag of a model is improved by manually analysing flow simulations and manually updating the model. This is a costly, time-consuming and labour-intensive process. Automated optimization of a model can help make the CAE-process both quicker and cheaper.

2 Aim

The main aim of this master's thesis is to deliver tools for the aerodynamic optimization of a vehicle's shape. The optimization process should be automated and need to be compatible with Volvo's computer aided engineering (CAE) process. A secondary aim is to provide an improved vehicle shape using the optimization tools. In summary the aim is to:

- develop tools for aerodynamic optimization of vehicle shape,
- automate optimization and
- provide an improved vehicle shape.

2.1 Limitations

The optimization is focused on minimizing the drag coefficient C_D. Consideration is taken to the projected area, to make sure the drag force ($F_D \sim C_D \cdot A$) does not increase even if the drag coefficient is lowered. The lift coefficient C_L is also taken into account, but other aspects of the aerodynamic shape such as noise, vibration etc. are not taken into account.

3 Theory

This chapter aims to give a short introduction to some of the concepts used in this thesis.

3.1 Aerodynamic forces and coefficients

A car driven on a road will experience aerodynamic forces acting on it, just as any bluff body moving through a fluid. There will be a drag force acting against the direction of travel and a lift force acting perpendicular to the road. For passenger cars this is generally a positive lift force acting to lift the car off the road. Due to side winds there will also be a force acting on the side of the car. The side force is neglected in this thesis. The drag and lift forces are computed by integrating the surface pressure p_s as

$$F_D = \int p_s \mathrm{d}A_x \text{ and} \tag{3.1}$$

$$F_L = \int p_s \mathrm{d}A_z, \tag{3.2}$$

where x is the direction of travel and z is perpendicular to the ground. These equations can be rewritten using dimensionless drag and lift coefficients as

$$F_D = \frac{C_D A \rho v^2}{2} \text{ and} \tag{3.3}$$

$$F_L = \frac{C_L A \rho v^2}{2}. \tag{3.4}$$

C_D is the drag coefficient, C_L is the lift coefficient, A is the projected frontal area, ρ is the density of air and v is the velocity. At subsonic speeds the density of air can be considered constant. For constant driving speed the relevant factors that influence the drag force are C_D and A. A larger frontal area will mean a larger drag force. Objects with the same frontal area can however experience different drag forces, which is why C_D can be used as a measure of how aerodynamic a certain bluff body is.

3.2 Design of experiments

With n design parameters there is an n-dimensional room of possible configurations. The amount of tested configurations should be kept to a minimum to avoid costly computations. A design of experiments method is a statistically reliable method to sample these configurations. A simple method is the full-factorial design where each parameter domain is divided into a number of intervals and each interval is sampled once. For i intervals and n parameters this results in i^n designs.

3.2.1 Latin Hypercube Sampling (LHS)

Latin hypercube sampling is possible for any number of dimensions but for simplicity the theory of a 2D LHS is presented here. In a 2D LHS, the sample space is divided into a number of rows and columns. Data is sampled exactly once from each row and each column. LHS has a smaller variance than a random sampling (Fang et al., 2006) and allows for a smaller number of samples than the full-factorial design. A simple LHS with four samples can be seen in Figure 3.1. The figure visualises the main concept of a LHS, where each row and each column has exactly one node.

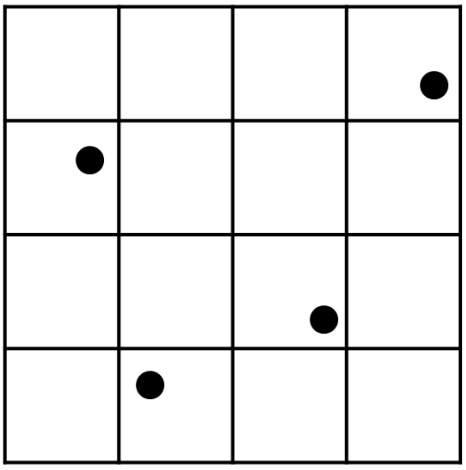


Figure 3.1: A latin hypercube sampling with four samples.

Maximin distance LHS

One of the drawbacks of LHS is that is does not reach a minimal variance (Fang et al., 2006). An improved model is the maximin distance LHS which aims to find an LHS where the minimum distance between points is maximized, i.e. where the points are as spread out as possible. This is done by adding a Monte-Carlo simulation to the LHS creation. A large number of LHS designs are generated and the minimum distance between any two points is computed for each design. The design with the largest distance is chosen as the final design. (Alam et al., 2004)

3.3 Artificial neural network

An artificial neural network is a computational model that mimics the behaviour of an organic nervous system made up of neurons and can among other things be used to approximate non-linear functions. The organic neurons are replaced with computational nodes. A common network is the *multi-layer feedforward network*. This network is generally made up of several layers of nodes; one input layer, one or more hidden layers and one output layer. Signals are transferred from the input layer to the output layer. Another possibility is a *recurrent network* where there is also feedback from the output to the input. Figure 3.2 shows a multi-layer feedforward network with one hidden layer. (Haykin, 2009)

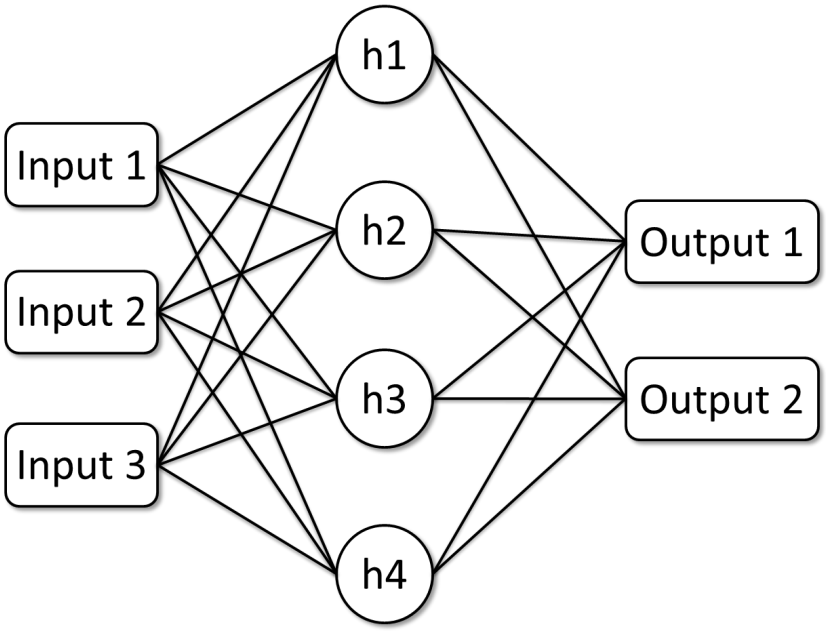


Figure 3.2: Schematic of an artificial neural network.

Consider the first input node in Figure 3.2. Data from this node is multiplied with a weight W_1 and transferred to the first node in the hidden layer. It is also multiplied with a different weight W_2 and transferred to the second node in the hidden layer and similarly for the hidden nodes three and four. Data is transferred in this manner between all nodes in the input layer and the hidden layer. At the first hidden node the weighted data from all input nodes is summed and to this sum is added a bias B_1. The same process is repeated for all hidden nodes. Data is transferred between the hidden layer and the output layer in the same manner as between the input layer and the hidden layer. (Van den Braembussche, 2008)

3.3.1 Initialising the network

The neural network is initialised by adjusting the weight and bias coefficients. This is known as 'training' and a common method is the *back-propagation method* (Van den Braembussche, 2008). A database of input values and corresponding output values are used to train the network. This database is generated from the original function or measured from the original system. The back-propagation method consists of two phases: a forward phase and a backward phase. In the forward phase weights and biases are kept fixed while a signal is transferred from the input layer and the output layer. In this way a set of responses are computed from the database input values. These responses are compared to the database output values. In the backward phase the resulting error signal is propagated backward through the network and weights and biases are adjusted. (Haykin, 2009)

3.3.2 Overtraining

When a neural network is trained with too large a database it may end up closely reflecting features of the training data (noise etc.) and lose its generalization ability. This is called overtraining. There are two main strategies for avoiding this; early stopping and regularization.

Early stopping

Using early stopping a sample of inputs and corresponding outputs are divided into two sets, a training set and a validation set. The neural network is trained repeatedly with the training set and the output error is computed. This error will decrease continuously. Periodically the training is stopped and the network is tested with the validation set. The output error is computed also for this set. The validation error will decrease at first when the generalization of the network is improved. After a while the network will start to become overtrained and the validation error will increase. By stopping the training at the point where the validation error is minimized overtraining is avoided. (Haykin, 2009)

Regularization

Regularization is based on the assumption that the approximated function is smooth, i.e. that there are no discontinuities in the physical response of the system. This assumption is valid for a subsonic fluid flow. A smooth response from the neural network is achieved by keeping the weights small. The goal when training a neural network is to minimize the output error E_T. Regularization aims to minimize the network weights E_W. This minimization problem can be stated as: (Foresee and Hagan, 1997)

$$E = \alpha E_T + \beta E_W. \tag{3.5}$$

The main problem with this is to find correct values for α and β. Foresee and Hagan (1997) presented a method of solving this optimization problem that provides good generalization. This method has been implemented in the MATLAB function *trainbr*.

3.4 Evolutionary optimization

Stochastic optimization algorithms that are inspired by natural evolution are generally called *evolutionary algorithms*. Methods inspired by the social behaviour of species, such as particle swarm optimization, are also included in this general term (Elbeltagi et al., 2005).

The origin of evolutionary algorithms is the genetic algorithm which mimics the natural evolution of species (Elbeltagi et al., 2005). The solution to a problem is encoded in a string of numbers. Each element in the string is the equivalent of a 'gene'. A number of initial solutions, or 'individuals', are created randomly to make up the first generation. New generations are created by combining the genes of the fittest individuals. Elements in some individuals are changed at random to simulate mutations. Generation by generation the solutions will become better and better and eventually reach a global optimum. (Goldberg, 1989)

3.4.1 Particle Swarm Optimization (PSO)

Kennedy and Eberhart (1995) suggested a method called particle swarm optimization that combines the global search of evolutionary algorithms with a local search. The method mimics the behaviour of a flock of birds searching for food. Initially the birds are spread out over an area. Each bird searches for food but also communicates with the other birds in the flock, moving towards the bird in the best position. In optimization terms this means a combination of a global search and a local search. Each particle, or bird, performs a local search. The particles are moving in the direction of the overall best solution, i.e. a global search.

3.4.2 Multiple objectives

Coello et al. (2004) proposed an extension to the algorithm by Kennedy and Eberhart (1995) that can handle multiple objectives. Instead of searching for a global optimum the algorithm searches for the Pareto front. They also incorporated a mutation step to improve the multi-objective search.

Pareto front

If the optimization problem has more than one objective there is generally not one single global optimum but a collection of optimal points called a Pareto front. A point is said

to be Pareto optimal if there is no way to improve the solution in one objective without making it worse in another. Mathematically this can be expressed as follows (Coello et al., 2004):

A tensor $\mathbf{u}$ is said to dominate $\mathbf{v}$ ($\mathbf{u}$,$\mathbf{v} \in \mathcal{R}^k$) if

$$\mathbf{u} \neq \mathbf{v} \text{ and } u_i \leq v_i \ \forall i \tag{3.6}$$

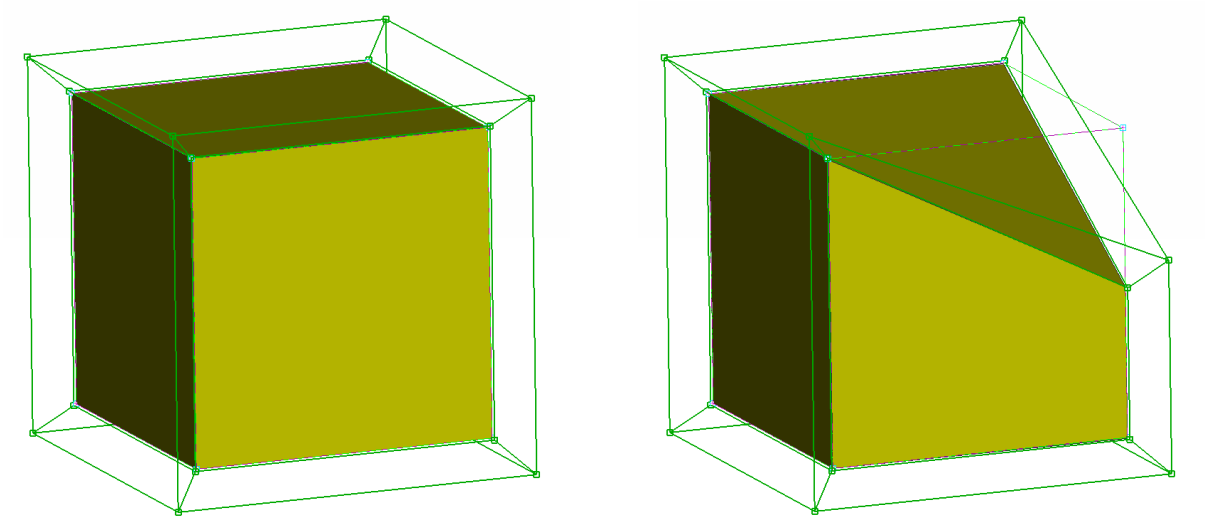
A point $\mathbf{x} \in \mathcal{R}^k$ with an objective function $\mathbf{f}(\mathbf{x})$ is nondominated if

$$\nexists \mathbf{x}' \in \mathcal{R}^k \text{ such that } \mathbf{f}(\mathbf{x}') \text{ dominates } \mathbf{f}(\mathbf{x}) \tag{3.7}$$

A point is called Pareto optimal if it is nondominated in the search space.

3.5 Mesh morphing

Mesh morphing is a method in which the mesh of a model can be updated, without having to change the original CAD model. In ANSA this is done by defining a number of morphing boxes which map to the surface of the model. When a morphing box is changed this will also change the mesh. An illustration of the morphing concept can be seen in Figure 3.3.



(a) Original box. (b) Morphed box.

Figure 3.3: Illustration of morphing concept. The outer green lines represent the morphing box. When the upper right corner of the morphing box is moved downwards the shape of the mesh is changed.

3.6 Turbulence model

Turbulence is the chaotic and irregular motion of fluid particles in a flow, for example in the flow around a car. The physical framework for describing fluid motion are the Navier-Stokes equations coupled with the continuity equation. It is possible to directly solve these equations using numerical methods, but for turbulent flows this requires a very high resolution (in both time and space) in order to resolve the smallest turbulent scales. For the flow around a car this is not a realistic approach at present day due to the high computational cost. Instead the turbulence is modelled to allow a lower resolution. A common approach is to use the Boussinesq assumption. Such turbulence models are called eddy viscosity models. When neglecting gravity and assuming steady-state incompressible flow the modelled continuity and Navier-Stokes equations can be written as (Davidson, 2013):

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0 \text{ and} \tag{3.8}$$

$$\rho \frac{\partial \overline{u}_i \overline{u}_j}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left[(\mu + \mu_t)\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)\right], \tag{3.9}$$

where μ_t is the turbulent viscosity defined in the Boussinesq assumption. μ_t is unknown and is computed using a turbulence model.

3.6.1 k-epsilon model

In the $k-\varepsilon$ turbulence model two additional equations are solved apart from the Navier-Stokes and continuity equations. These are the modelled transport equations for the kinetic energy k and its dissipation ε. They are used to compute the turbulent viscosity μ_t. The modelled transport equations for k and ε can be written as (Davidson, 2011)

$$\frac{\partial \rho \overline{u}_j k}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P_k - \rho\varepsilon \text{ and} \tag{3.10}$$

$$\frac{\partial \rho \overline{u}_j \varepsilon}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + \frac{\varepsilon}{k}(c_{\varepsilon 1} P_k - c_{\varepsilon 2}\rho\varepsilon), \tag{3.11}$$

where

$$P_k = \mu_t \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right)\frac{\partial \overline{u}_i}{\partial x_j}. \tag{3.12}$$

The turbulent viscosity is computed as

$$\mu_t = c_\mu \rho \frac{k^2}{\varepsilon}, \tag{3.13}$$

where c_μ is a constant. There are in total five constants in the $k-\varepsilon$ model (σ_k, σ_ε, $c_{\varepsilon 1}$, $c_{\varepsilon 2}$ and c_μ) which have been determined from experiments.

Realizable k-epsilon model

There are some problems with the standard $k-\varepsilon$ model, mainly to do with the modelled ε equation and the formulation of μ_t. Shih et al. (1995) proposed the *realizable* $k-\varepsilon$ *model* as an improvement over the standard formulation. This model includes a new formulation of the modelled ε equation and a new formulation of the turbulent viscosity μ_t based on the realizability constraints. The realizability constraints assure that the flow is physically realizable, something that is not always the case for the standard $k-\varepsilon$ model. The realizability constraints are stated as (Davidson, 2013):

$$\overline{u_\alpha'^2} \geq 0 \quad \forall \alpha \in \{1,2,3\} \text{ and} \tag{3.14}$$

$$\frac{|\overline{u_\alpha' u_\beta'}|}{\left(\overline{u_\alpha'^2}\ \overline{u_\beta'^2}\right)^{1/2}} \leq 1 \quad \forall \alpha,\beta \in \{1,2,3\}. \tag{3.15}$$

3.7 Statistical concepts

Variance is a measure of the variation in a group of data points. The variance of a variable is the average of the squared deviation of the variable from its mean (Rice, 2007):

$$Var(X) = E[(X - \mu_X)^2]. \tag{3.16}$$

E denotes the expectation value and μ_X is the mean of X. Standard deviation is a more commonly used concept than variance. Standard deviation (σ) is defined as the square root of the variance:

$$\sigma_X = \sqrt{Var(X)}. \tag{3.17}$$

Standard deviation can be used to express a confidence interval. For a normal distribution the 95% confidence interval is $\pm 2\sigma$, i.e. a data point in X is with 95% certainty located in the range $\mu_x \pm 2\sigma_X$ (Rice, 2007).

Covariance is a measure of how two variables are associated. It is defined as (Rice, 2007)

$$Cov(X,Y) = E[(X - \mu_X)(Y - \mu_Y)]. \tag{3.18}$$

$X-\mu_X$ and $Y-\mu_Y$ are the deviation of X and Y from their mean values and the covariance is the average value of this product. If X and Y are independent the covariance will be zero, i.e. there is no association between the variables. But if for example X is larger than its mean when Y is also larger than its mean this will give a positive covariance. A similar measure is the correlation coefficient (Rice, 2007)

$$\rho = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}, -1 \leq \rho \leq 1. \tag{3.19}$$

The correlation coefficient is a dimensionless value between -1 and 1 that measures the quality of a linear relation between X and Y. A value of 1 means a perfectly linear positive relation. A value of -1 means a perfectly linear negative relation.

4 Method

This chapter presents the method used in this thesis.

4.1 CAE process

The optimization tool was designed to be compatible with Volvo's computer aided engineering (CAE) process. This process involves four steps with different software, see Figure 4.1, where the optimization is included as a fifth step.

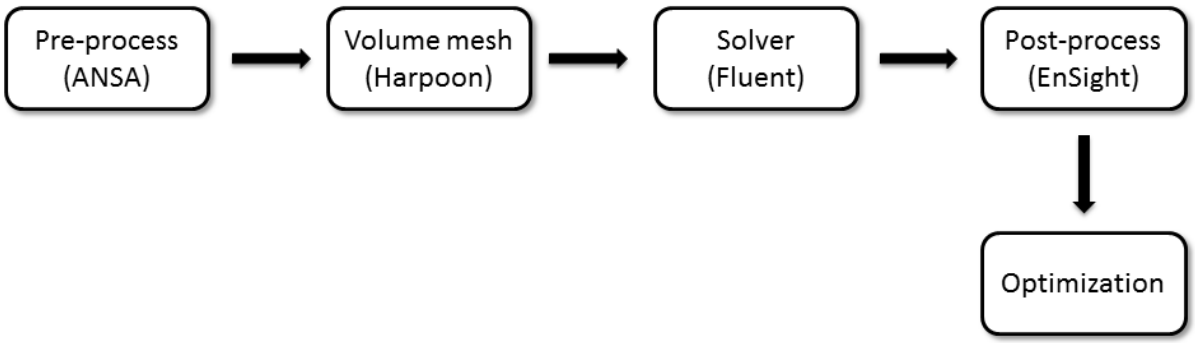


Figure 4.1: Schematic of CAE process.

4.1.1 Pre-processing

Any geometry cleaning and mesh adjustments are applied manually in ANSA before generating a surface mesh. Mesh morphing is executed and the mesh is exported. The cell size of the surface mesh is not very relevant since a new mesh is generated in Harpoon.

4.1.2 Volume meshing

A volume mesh is generated automatically in Harpoon from the ANSA mesh file. The cell sizes vary according to a number of refinement boxes, see Figures 4.2 and 4.3. The

cell sizes in each refinement box are given in Table 4.1. Cell sizes outside and between refinement boxes are expanded continuously.

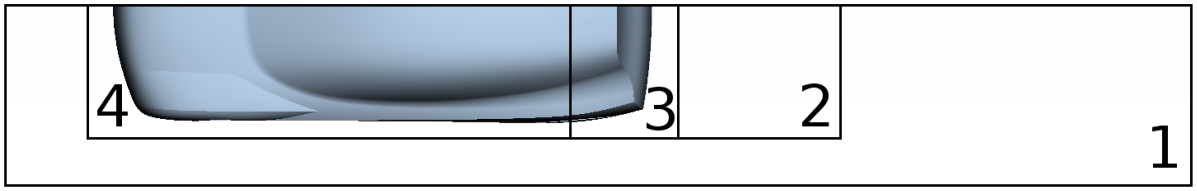


Figure 4.2: Top view of refinement boxes.

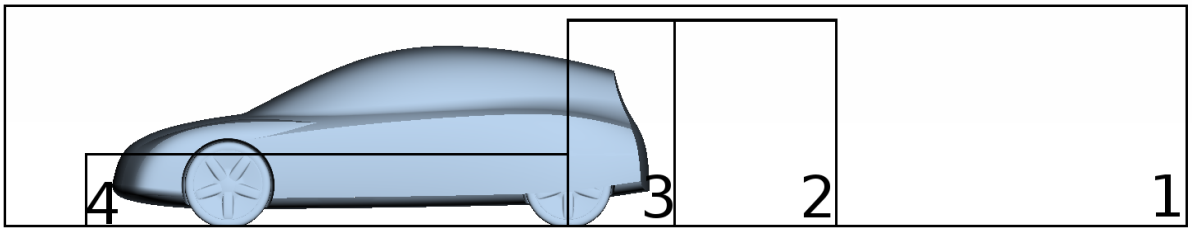


Figure 4.3: Side view of refinement boxes.

Table 4.1: Cell sizes in volume mesh. Default value of base level is 40 mm.

Area	Cell size
1. Around car	(Base level)
2. Wake	(Base level)/2
3. Wake	(Base level)/4
4. Under car	(Base level)/4

The cell sizes are controlled through a base level parameter. The default value is 40 mm but this can be adjusted.

4.1.3 Solver

The flow is solved with a steady-state pressure-based Navier-Stokes solver in Fluent. A realizable $k - \varepsilon$ turbulence model is used together with a second-order discretization scheme for pressure and momentum and a first-order discretization scheme for k and ε. The solver is first run 1000 iterations with Fluent's default under-relaxation factors, before the factors for pressure and momentum are lowered to 0.3 and further iterations are run. The air velocity is set to 100 km/h, simulating a car driving at highway speeds.

Comparison of turbulence models

The two-equation eddy viscosity models are not as accurate as more advanced models such as RSM or LES but they are a good trade-off between accuracy and speed and therefore the most commonly used models in the automotive industry. The standard $k-\varepsilon$ is a common and well-documented approach, but it behaves poorly in regions where there is an adverse pressure gradient or a low Reynolds number (Davidson, 2011). Singh et al. (2005) compared the performance of four turbulence models (Spalart-Allmaras, $k-\varepsilon$, RNG $k-\varepsilon$ and realizable $k-\varepsilon$) for the flow around a truck and found that realizable $k-\varepsilon$ best matched the experimental results.

Solver accuracy

In this thesis C_D and C_L are reported with 3 decimals and even higher accuracy is used for computations. This accuracy concerns the convergence of the solver. It does not entail that the accuracy of the simulation itself, as compared to the real physical problem, is this high. The simulation accuracy is probably much lower. However, since a neural network will be used to approximate the solver it is important that the simulations are converged and that the relative changes between simulations are accurate.

4.1.4 Post-processing

Post-processing is performed in EnSight where a number of different plots are generated.

4.2 Aerodynamic Concept Car (ACC)

The Aerodynamic Concept Car is a concept model developed at Volvo. It is a simplified model but still captures the main aspects of a real car. The ACC can be seen in Figures 4.4 and 4.5.

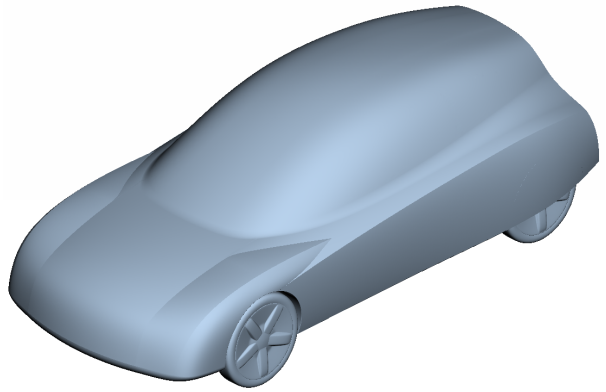
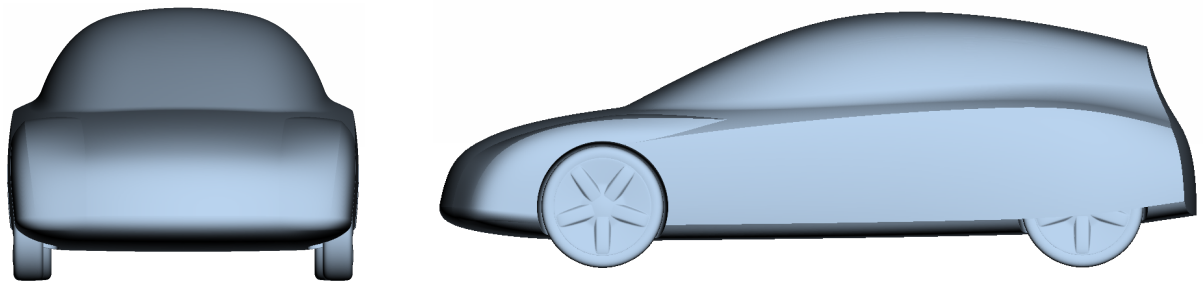


Figure 4.4: Aerodynamic Concept Car (ACC).



(a) Front view.

(b) Side view.

Figure 4.5: Front and side view of ACC.

It is designed to be a low-drag model with realistic proportions and is therefore a good test object for an optimization study. If the optimization tool is able to lower the drag of this model which has been developed to have an optimal aerodynamic shape, it should be able to do so also for a model which has not.

4.3 General optimization method

There are a number of different methods for optimizing the aerodynamic shape of a vehicle but they all share some common aspects. The schematics of a general method for optimizing the aerodynamic shape of an object can be seen in Figure 4.6.

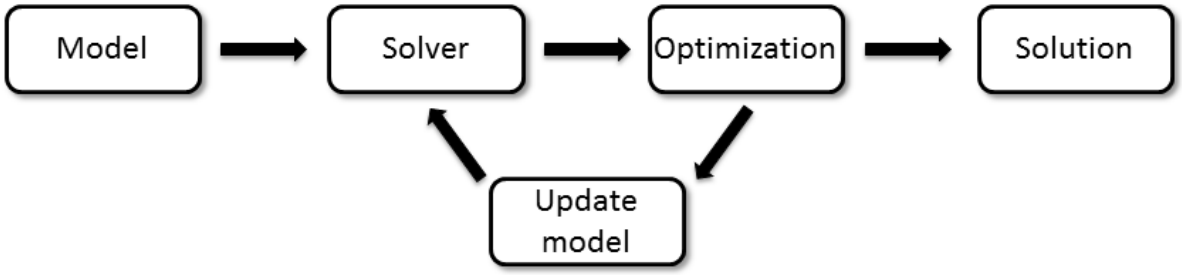


Figure 4.6: Schematics of general method for car shape optimization.

The method starts with a single initial model. A CFD simulation is performed for this model. The result is analysed using some optimization algorithm. Some design parameters of the model are updated and a new CFD simulation is performed on the updated model. This iterative process continues until some stop condition is reached. This method is adopted in Helgason and Hafsteinsson (2009), Kim et al. (2009) and Dumas (2008). The method is successfully implemented in all of the three reports. It is however very computationally demanding and they have all limited their studies to simplified models or coarse meshes. The large computational cost associated with this method limits the real-world applications for the automotive industry at present day.

4.4 Two-level optimization

A way to reduce the computational cost is to perform the optimization using an approximation of the solver. This approach is sometimes called *two-level optimization*. Van den Braembussche (2008) reports that the use of such an optimization can considerably decrease the computation time. A schematic of the method described by Van den Braembussche (2008) can be seen in Figure 4.7.

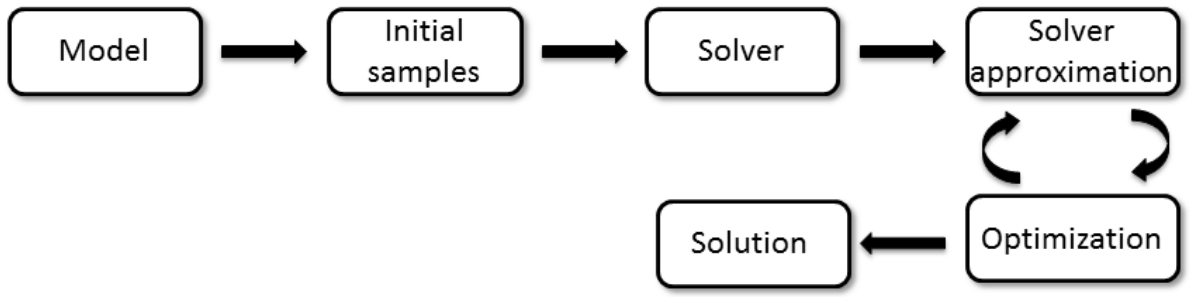


Figure 4.7: Schematics of two-level car shape optimization.

Instead of performing a new CFD simulation every time the model is updated, initial simulations are performed for a sample of the configuration space. The samples are chosen using a design of experiments method. The results from these initial simulations are used to create an approximative model of the aerodynamic behaviour. Optimization is performed on this faster approximative model. This approach is used successfully in Lietz (2011) and Song et al. (2012). There are a number of method choices involved in these processes; the choice of a solver approximation method, a design of experiments method and an optimization algorithm.

4.4.1 Solver approximation method

In the two-level optimization a number of CFD-simulations with different configurations of design parameters are performed. The results of these simulations are used to create an approximate relationship between the aerodynamic behaviour and the design parameters. This approximation can be done with a number of methods. Two common methods are response surfaces and artificial neural networks. Artificial neural networks can better capture non-linear effects than response surfaces (Song et al., 2012). Due to the complex nature of the optimization problem at hand non-linear effects are a possibility and therefore an artificial neural network was adopted as the solver approximation method in this thesis.

Layers and nodes

Song et al. (2012) used a neural network for a shape optimization study of a car with six parameters and found that the best accuracy was achieved for one layer of hidden nodes with 1.5 to 2 times more hidden nodes than parameters. A single layer with two times more hidden nodes than parameters was used in this thesis.

Sample size

Which sample size is required for a reliable approximation using neural networks is hard to know beforehand. Song et al. (2012) used 64 samples to train their neural networks and was able to accurately predict C_D. In this thesis a sample size N was chosen according to Song et al. (2012) as:

$$N = 2^m, \tag{4.1}$$

where m is the number of parameters.

Training

The neural networks were trained with the commonly used Levenberg-Marquardt back-propagation method. The mean square of the output errors was used to measure the training error. Regularization was used to avoid overtraining since this allows use of all samples for training. For small databases regularization yields better generalization than early stopping (Beale et al., 2014).

4.4.2 Design of experiments

Van den Braembussche (2008) found that the accuracy of a neural network is increased for a fixed number of samples when the samples are chosen using a design of experiments method instead of being random. Alam et al. (2004) investigated a number of different design of experiments methods used in conjunction with an artificial neural network. The study included a full factorial sampling, a random sampling, a central composite sampling and a latin hypercube sampling. They found that the latin hypercube sampling performed best of the tested methods. Latin hypercube sampling was adopted in this thesis.

4.4.3 Optimization algorithm

Finally an optimal configuration of design parameters is determined using an optimization algorithm. There are many different ways to find an optimal solution, ranging from

traditional gradient-based algorithms to more modern evolutionary algorithms. Gradient-based algorithms are in general faster than evolutionary methods but do not as reliable find a global optimum. They risk being trapped at local optima (Giannakoglou and Papadimitriou, 2008). Evolutionary methods are on the other hand robust but they are time-consuming and do not have as fine a convergence as gradient-based methods (Muyl et al., 2004).

There are many different methods suggested to improve the accuracy of genetic algorithms by combining them with a local search, for example the particle swarm optimization suggested by Kennedy and Eberhart (1995). Elbeltagi et al. (2005) compared the success rate and solution quality of five different evolutionary methods: genetic, memetic, particle swarm, ant-colony and shuffled frog leaping algorithms. They found that particle swarm optimization performed best.

Coello et al. (2004) extended the particle swarm optimization (PSO) algorithm to allow for multiple objective search. This multi-objective particle swarm optimization (MOPSO) was compared with three evolutionary algorithms known to perform well in multi-objective optimization problems (NSGA-II, PAES and microGA). The MOPSO algorithm was found to perform well and to be computationally cheap.

The PSO algorithm used in this thesis was adapted from Elbeltagi et al. (2005) and Coello et al. (2004). A summary of the algorithm is presented below.

Summary of PSO algorithm

1. Create a population of particles with random positions in domain
2. For each generation:
   (a) Use neural network to compute C_D of each particle
   (b) Choose best particle as leader
   (c) Move other particles towards leader
   (d) Mutate position of some particles
3. Return best particle in last generation as solution

The MOPSO algorithm used in this thesis was adapted from Coello et al. (2004). A summary of the algorithm is presented below.

Summary of MOPSO algorithm

1. Create a population of particles with random positions in domain
2. For each generation:
   (a) Use neural network to compute C_D of each particle
   (b) Find Pareto front

   (c) Choose leaders from Pareto front (each particle may have a different leader)

   (d) Move particles towards their respective leaders

   (e) Mutate position of some particles

3. Return Pareto front in last generation as solution

4.4.4 Implementation

The optimization tool is divided into two separate parts: database generation and optimization.

Database generation

The database generation is executed through a custom Python script called *generateDatabase*. The first part of the code is a design of experiment, i.e. generation of all design configurations. This is done using a maximin distance latin hypercube sampling (LHS). For each configuration a mesh is created by morphing the original mesh in ANSA. Then the CAE process described in section 4.1 is executed, meaning that the morphed mesh is exported from ANSA and a volume mesh is generated by Harpoon. A simulation is run in Fluent to determine C_D and C_L. Finally all configurations and corresponding C_D and C_L values are stored in a database file.

Summary of *generateDatabase*

1. Generate configurations

   (a) Create large number of LHS designs

   (b) Compute minimum distance between any two points in each LHS

   (c) Pick LHS design with largest minimum distance

2. For each configuration:

   (a) Morph mesh

   (b) Generate volume mesh

   (c) Run Fluent simulation

   (d) Compute C_D and C_L

3. Write database file with configurations, C_D and C_L

Optimization

The optimization is executed through a custom Python script called *runOptimization*. First the database created by *generateDatabase* is loaded. An artificial neural network is created and trained with this database. This is done in Matlab using the *Neural Network Toolbox*. The purpose of the neural network is to predict C_D or C_L for new unknown configurations, i.e. to act as an approximation for a simulation in Fluent. When the network is trained an optimal configuration is determined using particle swarm optimization. A parameter study is carried out in order find the effect of each parameter. Finally a simulation is run in Fluent for the optimal configuration.

Summary of *runOptimization*

1. Load database
2. Create neural network
   - single layer of hidden nodes
   - twice the number of hidden nodes as parameters
3. Train neural network (using *trainbr* in Matlab)
   - Levenberg-Marquardt back-propagation
   - Bayesian regularization
4. Find best configuration(s) using PSO or MOPSO
   - If MOPSO, choose one or more desired configurations from Pareto front
5. Run parameter study
6. Run simulation for optimal configuration(s)
   (a) Morph mesh
   (b) Generate volume mesh
   (c) Run Fluent simulation

4.5 Method validation

Before commencing the full model optimization study a smaller study was executed to validate the optimization tool and to evaluate the performance of the custom code implementation (*generateDatabase* and *runOptimization*, see section 4.4.4). The performance of *runOptimization* was compared to that of the commercial software modeFrontier. This study aimed to lower C_D and consisted of four samples and the baseline configuration. Two parameters were studied; roof drop and boat tailing. For further details regarding the parameters, see sections 4.7.1 and 4.7.4. The roof drop varied between -20 mm to

+60 mm from the baseline and the boat tailing between +0 mm to +50 mm from the baseline. A simulation with 2000 iterations was performed for each sample to determine C_D. Since the focus was validation of the optimization no greater consideration was put into solver details at this stage and the default settings in the CAE process was used, see section 4.1.

The configurations and corresponding simulation results were used to create a database for the optimization. In modeFrontier the size of the neural network was determined automatically. *runOptimization* used one layer with four nodes for the neural network. In both cases all samples were utilised for training the network. Both cases also used 50 particles and 100 generations for the particle swarm optimization. Confirmation simulations were run for the predicted optimal solutions.

4.6 Convergence study

A study was performed on the baseline case to determine the number of iterations required for convergence and to evaluate the solver settings. The study was focused on testing iteration and grid independence. The default settings in the CAE process was used for most parameters (see section 4.1) but the number of iterations, the cell size and the under-relaxation factors of pressure and momentum were varied. The study covered 10000 iterations for a grid of 12.9 million cells (*base level* 40 mm) where a number of different under-relaxation factors were evaluated. Finally a simulation on a finer grid of 24.1 million cells (*base level* 30 mm) was performed using the best choice of under-relaxation factors. This simulation covered 5000 iterations.

4.7 Model optimization study

A large study of five design parameters was executed with the aim of lowering C_D of the ACC model described in section 4.2. The study consisted of 33 configurations (including the baseline configuration). The number of configurations was chosen as 2^m, where m is the number of parameters. The optimization was performed using the two scripts *generateDatabase* and *runOptimization* described in section 4.4.4.

The configurations were generated using a latin hypercube sampling. The minimum distance between any two points was maximized by generating 10000 different LHS designs and choosing the design with the largest minimum distance.

The default settings in the CAE process was used for most parameters, see section 4.1, but the under-relaxation factors of pressure and momentum were set to the optimal value 0.3 determined in the convergence study. 4000 iterations were run with a grid of 12.9 million cells. C_D was computed as the average of the last 2000 iterations.

A neural network with a single layer of ten hidden nodes was used to approximate C_D in the optimization step. The optimal configuration was determined using a particle swarm

optimization with 50 particles and 100 generations.

The five studied design parameters were roof drop, underbody lift, diffuser lift, boat tailing and front wheel cover. These parameters varied between the baseline (minimum) values and the fully morphed (maximum) values defined in Table 4.2. The limits were chosen as the maximum changes that would still produce a realistic car.

Table 4.2: Design parameter limits.

Parameter	Baseline [mm]	Maximum [mm]
Roof drop	0	100
Underbody lift	0	107
Diffuser lift	0	100
Boat tailing	0	50
Front wheel cover	0	60

The mesh changes were implemented with mesh morphing in ANSA using a system of morph boxes depicted in Figure 4.8.

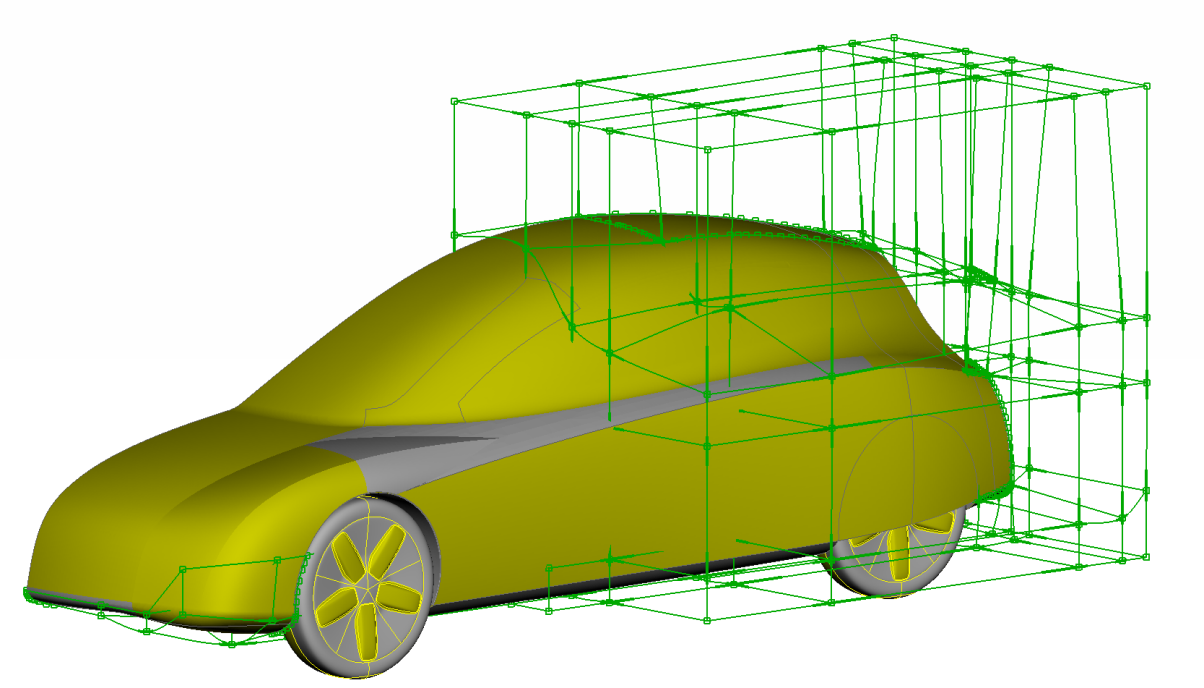


Figure 4.8: Half car model with morph boxes.

The mesh morphing changed the projected frontal area for some configurations but this change was always a decrease in area. This is important so that a decrease in the drag coefficient always corresponds to a decrease in drag force ($F_D \propto C_D \cdot A$).

4.7.1 Roof drop

A roof drop was achieved by compressing a portion of the roof from the highest point to the rear end. The compression had a linear variation with full compression at the rear end and no compression at the highest point of the roof. A comparison of the baseline and the maximum roof drop of 100 mm can be seen in Figure 4.9.

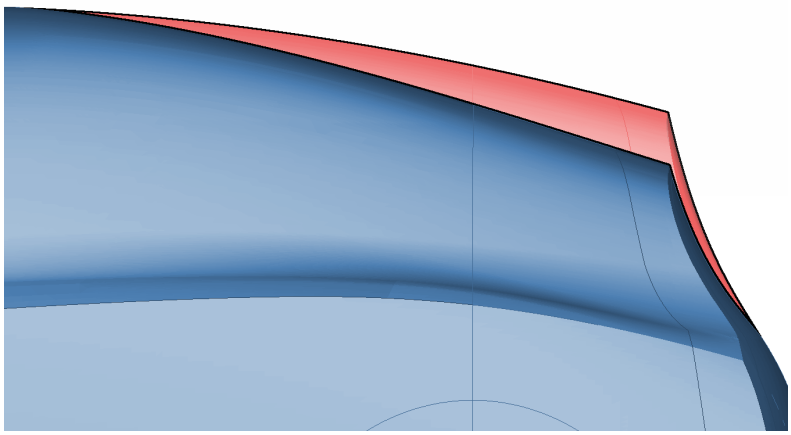


Figure 4.9: Comparison of baseline and fully morphed roof. Baseline coloured red in background and fully morphed coloured blue in foreground. Maximum roof drop is 100 mm.

4.7.2 Underbody lift

The baseline underbody has a curvature along the car. The underbody was lifted to achieve a flatter profile. The maximum underbody lift was 107 mm, at which point the underbody was flat.

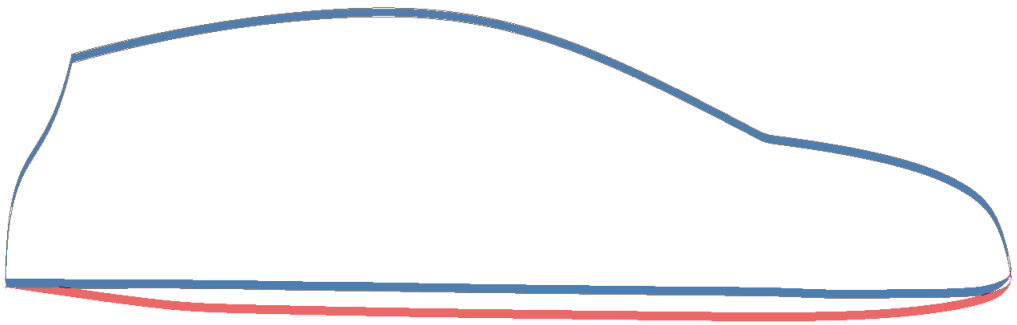


Figure 4.10: Comparison of baseline and fully morphed underbody contours. Baseline is the lower line coloured in red and fully morphed is the upper line coloured in blue.

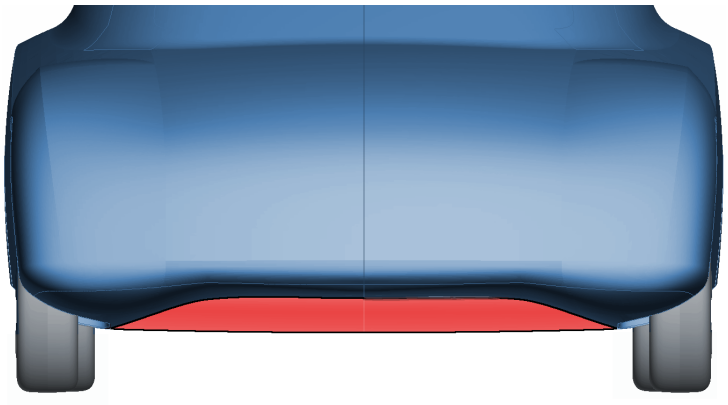


Figure 4.11: Comparison of baseline and fully morphed underbody as seen from the front. Baseline coloured red in background and fully morphed coloured blue in foreground. Maximum underbody lift is 107 mm.

4.7.3 Diffuser lift

The diffuser was changed by moving the rear edge upwards by a maximum distance of 100 mm.

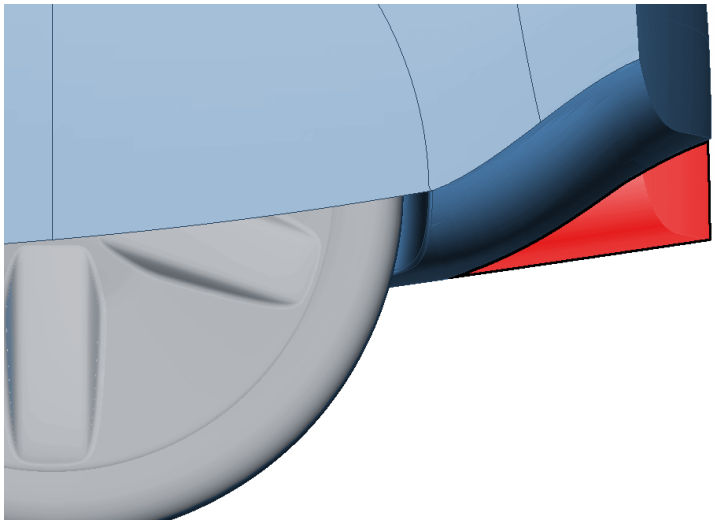
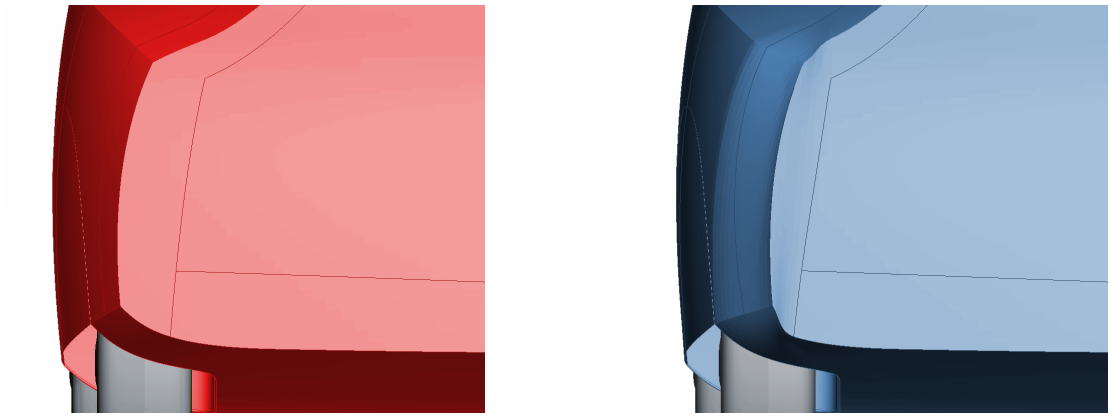


Figure 4.12: Comparison of baseline and fully morphed diffuser. Baseline coloured red in background and fully morphed coloured blue in foreground. Maximum diffuser lift is 100 mm.

4.7.4 Boat tailing

The rear end was slimmed by moving the rear side edge of the car inwards. This is known as boat tailing. A comparison of the baseline and the maximum boat tailing of 50 mm can be seen in Figure 4.13.

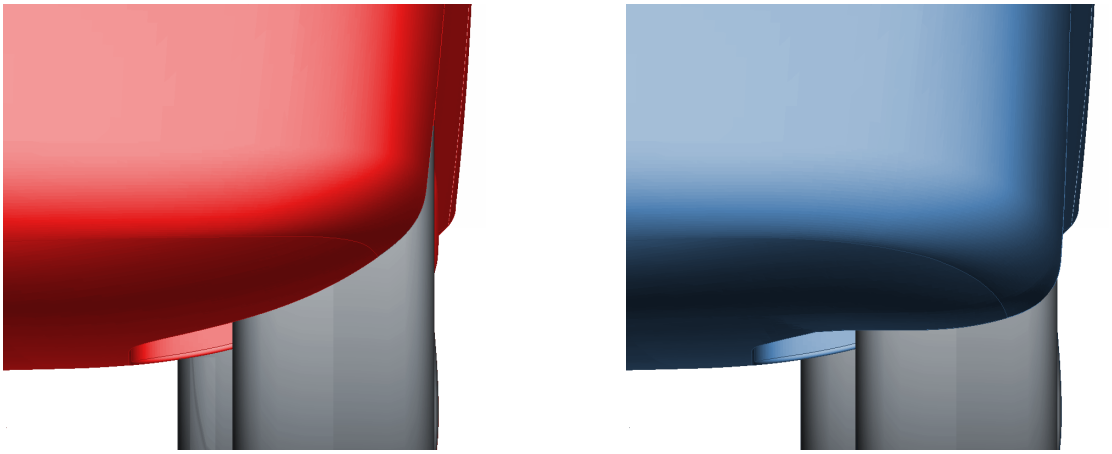


(a) Baseline. (b) Fully morphed.

Figure 4.13: Comparison of baseline and fully morphed rear. In the fully morphed case a boat tailing is applied, i.e. the rear is slimmed by moving the rear edge inwards. The maximum boat tailing is 50 mm.

4.7.5 Front wheel cover

An increased front wheel cover was achieved by extending the front wheel cover both outwards and downwards. The outwards change was fixed as 1/6 of the downwards change. The maximum extension was 60 mm downwards and 10 mm to the side.



(a) Baseline. (b) Fully morphed.

Figure 4.14: Comparison of baseline and fully morphed front wheel cover. Maximum wheel cover is extended 10 mm to the side and 60 mm downwards.

4.8 Considering lift

The model optimization study described above was focused on lowering the C_D of the ACC model. This study was extended with a second optimization study with the aim of

lowering both C_D *and* C_L, i.e. a multi-objective optimization. The database from the first study was also used in the second study.

Two similar neural networks, each with a single layer of ten hidden nodes, was used to approximate C_D and C_L. The Pareto front of configurations was determined using a multi-objective particle swarm optimization with 100 particles and 200 generations.

4.9 Database size

Four different sizes of the database were tested (4, 8, 16 and 32 elements) in order to determine how sensitive the prediction of the neural network is to database size. For each test a new database was constructed by random selection from the original database. This database was used to train the neural network. C_D and C_L were predicted for five control points and the error for each point computed. Ten tests were performed for each of the database sizes and the average error over these tests was computed.

4.10 Summary

An optimization tool fully compatible with Volvo's CAE process was developed. The optimization is a type of two-level optimization, meaning that a solver approximation is first created and then the optimization is performed using that approximation. A small study was performed to validate the optimization tool. A convergence study was also performed to find the optimal settings for the CAE process.

In short the optimization tool works by first generating a number of vehicle shape configurations using latin hypercube sampling. A new mesh is created for each configuration by morphing the original mesh. A flow simulation is run on each mesh in order to determine the drag and lift coefficients. When all simulations are finished the configurations and the resulting C_D and C_L are stored in a database. The database is used to train a neural network as an approximation to the simulations. A particle swarm optimization is used on the neural network to find the minimal drag and lift coefficients. Finally a simulation is performed on this optimal configuration to confirm the drag and lift prediction.

The main part of this thesis was a large study of five design parameters with the aim of lowering C_D of the ACC model. The study consisted of 32 configurations as well as the baseline configuration. The five studied parameters were roof drop, underbody lift, diffuser lift, boat tailing and front wheel cover. The study was extended with a multi-objective optimization with the aim of lowering both C_D and C_L.

Finally the prediction error of the neural network was computed for different sizes of the database.

5 Result

This chapter presents the results from simulations and studies carried out in this thesis.

5.1 Method validation

A small study of five cases was performed to validate the optimization tool. The configurations and corresponding drag coefficients are presented in Table 5.1. The baseline configuration is presented at the top of the table.

Table 5.1: Configurations and results from method validation study. Configurations defined as variation from baseline.

Roof drop [mm]	Boat tailing [mm]	C_D
0.0	0.0	0.167
57.3	21.2	0.157
17.6	31.8	0.163
-11.5	45.5	0.173
32.2	1.3	0.161

5.1.1 Optimization with modeFrontier

The predicted best configuration is a roof drop of 60.0 mm (max) and a boat tailing of 0.0 mm (min) with a corresponding C_D of 0.157. A confirmation simulation was performed and resulted in a C_D of 0.159. The computation time was 199 s (optimization step only).

5.1.2 Optimization with custom code *runOptimization*

The predicted best configuration is a roof drop of 60.0 mm (max) and a boat tailing of 17.8 mm with a corresponding C_D of 0.156. A confirmation simulation was performed and resulted in a C_D of 0.156. The computation time was 7 s (creation and training of ANN as well as optimization).

5.2 Convergence study

A convergence study was performed on the baseline case to evaluate the solver settings. A number of different under-relaxation factors for pressure and momentum were evaluated in simulations with 10000 iterations for a grid of 12.9 million cells. This corresponds to a base level in Harpoon set to 40 mm. The solution converged after about 2000 iterations for all cases but with oscillations of different magnitude. The standard deviation of C_D in the converged solution (i.e. iterations 2000-10000) was computed as a measure of these oscillations, see Figure 5.1.

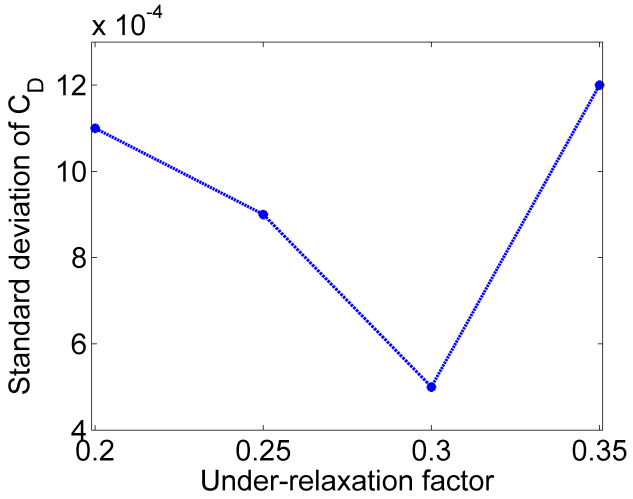
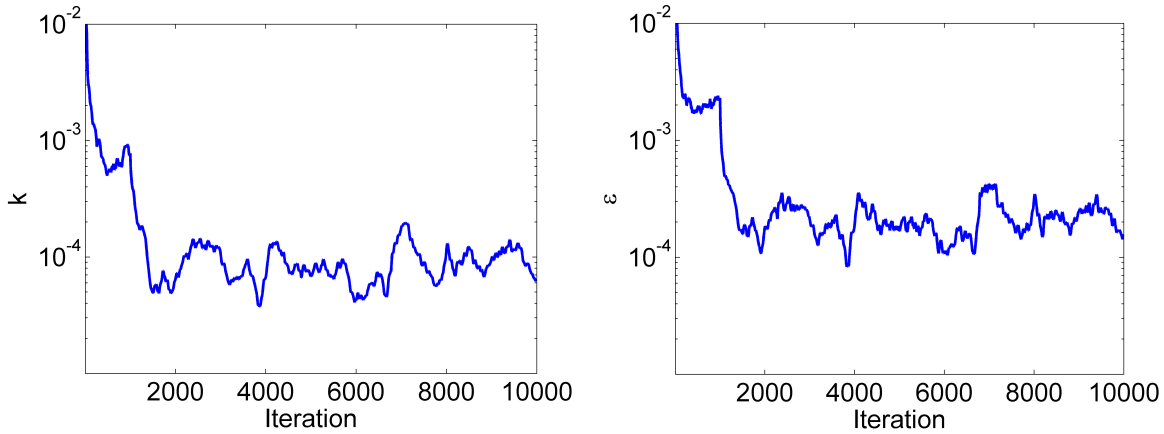


Figure 5.1: Standard deviation of C_D for different under-relaxation factors.

As can be seen in Figure 5.1 the best precision is achieved with the under-relaxation factors for pressure and momentum set to 0.3. Figure 5.2 show the residuals of k and ε over 10000 iterations for this case.



(a) Residual of k.

(b) Residual of ε.

Figure 5.2: Residuals of k and ε over 10000 iterations for simulation on grid with 12.9 million cells.

The residuals have decreased by more than three orders of magnitude, almost four, which indicate convergence. As for k and ε the solution of C_D appears converged but there are some oscillations, see Figure 5.3.

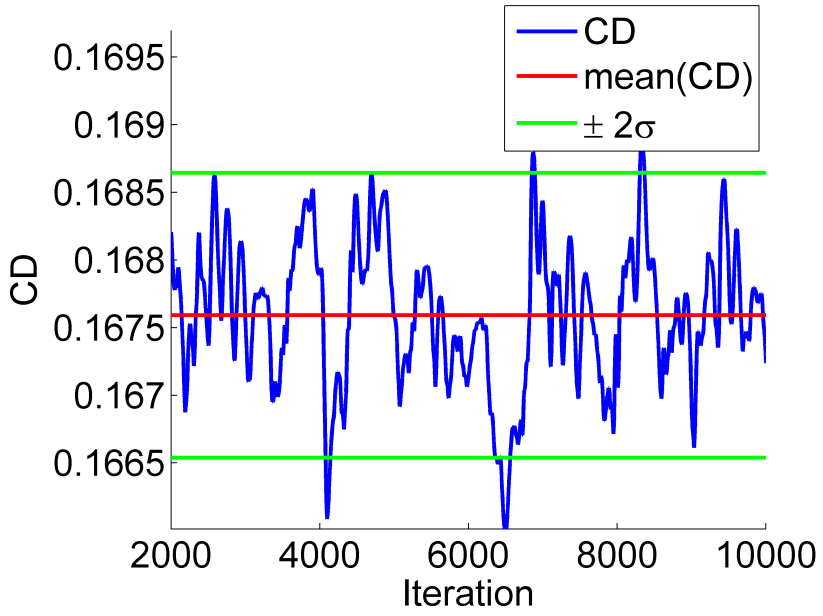


Figure 5.3: Variation of C_D over 10000 iterations for simulation on grid with 12.9 million cells.

The mean value of C_D (iterations 2000-10000) is 0.168 with a standard deviation σ of 0.0005. This means that the 95% confidence interval of C_D is

$$C_D = 0.168 \pm 0.001, \tag{5.1}$$

i.e. the variation of C_D from the mean value is less than one thousand. This confidence interval concerns the convergence of the solver. It does not entail that the accuracy of the simulation itself, as compared to the real physical problem, is this high.

A second simulation using the same under-relaxation factors was performed with a finer grid of 24.1 million cells. This corresponds to a base level in Harpoon set to 30 mm. The simulation was run for 5000 iterations. The oscillations are larger with the finer grid which shows in the standard deviation. The mean value of C_D (iterations 2000-10000) is 0.161 with a standard deviation σ of 0.0015. This means that the 95% confidence interval of C_D is

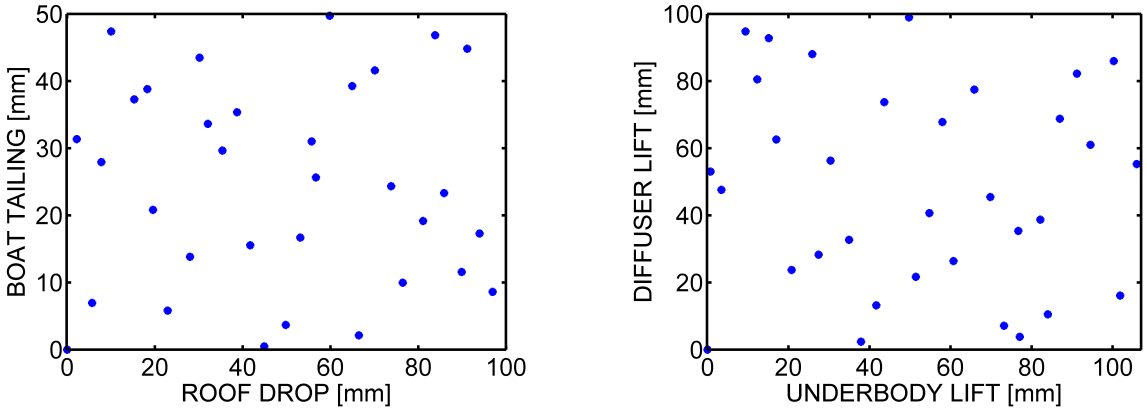
$$C_D = 0.161 \pm 0.003. \tag{5.2}$$

5.3 Model optimization study

The model optimization study covered five design parameters and a database of 33 simulations. The optimal configuration lowered C_D with 12.6%.

5.3.1 Sampling

The minimum distance between any two points in the latin hypercube sampling is 31.1 mm. If instead a full factorial design with the same number of samples had been used (i.e. two points spread out evenly for each parameter plus the baseline) the minimum distance had been less than 25 mm. Since the configuration space is 5-dimensional it is hard to visualise but as an example the 2D configuration space for two planes are shown in Figure 5.4.



(a) Plane 1: Roof drop and boat tailing.

(b) Plane 2: Underbody lift and diffuser lift.

Figure 5.4: 2D visualisation of configuration space plotted on two different planes.

5.3.2 Simulation convergence

The standard deviation of C_D was computed for all simulations in order to make sure the simulations converged. The maximum standard deviation is $6.9 \cdot 10^{-4}$ which corresponds to a 95% confidence interval with a deviation of 0.0014 from the mean value. However, the standard deviation is only this large in a couple of cases. The average standard deviation is $2.0 \cdot 10^{-4}$.

5.3.3 Optimization convergence

The minimum C_D in each generation of the particle swarm optimization is plotted in Figure 5.5.

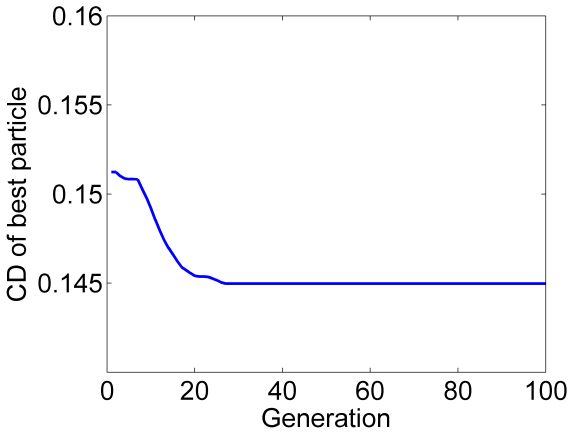


Figure 5.5: Minimum C_D in each generation for particle swarm optimization.

The figure shows that the optimization is converged after about 25 generations. The standard deviation of C_D for the last 50 generations is less than 10^{-10}. The optimization was also run with 100 particles and 1000 generations. It was run several times with both these and the original settings, each time reinitialising the neural network. All runs yielded the same result. Since the neural network was reinitialised every time, this indicates that the neural network is also converged.

5.3.4 Optimal configuration

The simulation of the baseline configuration resulted in a C_D value of 0.167. The optimal configuration is predicted to have a C_D of 0.145. This configuration is shown in Table 5.2.

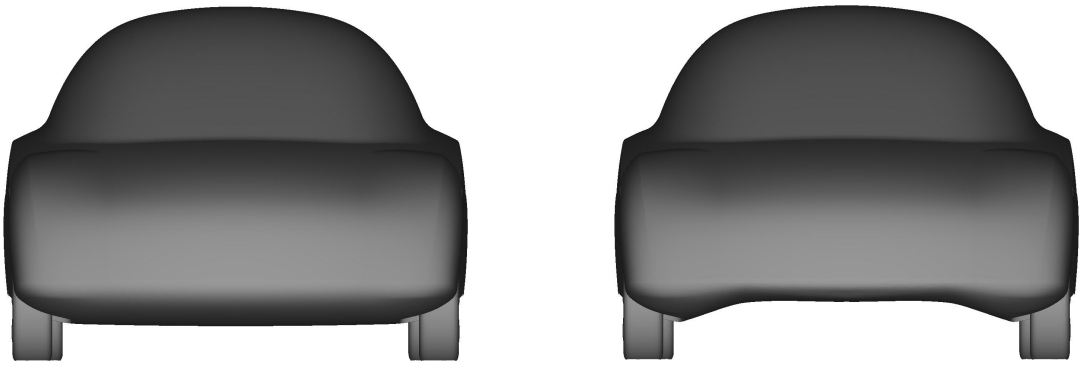
Table 5.2: Predicted optimal configuration.

Parameter	Configuration [mm]
Roof drop	100.0 (max)
Underbody lift	107.0 (max)
Diffuser lift	0.0 (min)
Boat tailing	31.3
Front wheel cover	0.0 (min)

A confirmation simulation was run for the optimal configuration. This simulation yielded a C_D value of 0.146, which corresponds to an improvement in C_D from the baseline configuration of 12.6%. The exact deviation of the predicted C_D from the simulation value is 0.0005, a deviation which is of the same order of magnitude as the simulation standard deviation. Expressed as a relative quantity the deviation of the predicted C_D from the simulation value is 0.3%. This is the prediction error of the optimization as compared to the CFD simulation; it says nothing about the error in the CFD simulation itself.

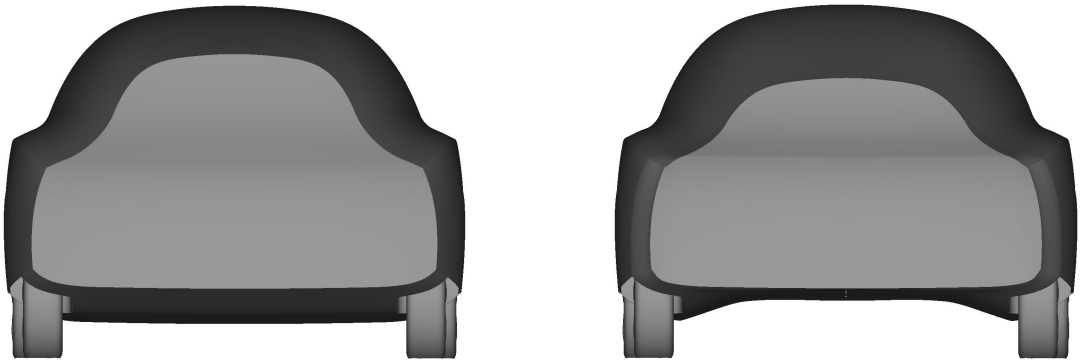
Visual comparison of baseline and optimal configurations

A 3D comparison of the baseline and optimal model is shown from different angles in Figures 5.6 to 5.8.



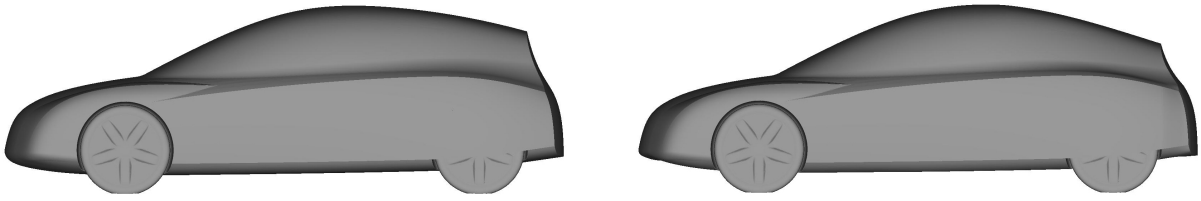
(a) Baseline. (b) Optimal.

Figure 5.6: Comparison of car front between the baseline and optimal configurations. The figures show the difference in underbody between the models.



(a) Baseline. (b) Optimal.

Figure 5.7: Comparison of car rear between the baseline and optimal configurations. The underbody lift and roof drop of the optimal model can be seen, as well as the boat tailing. The rear area of the optimal model is visibly smaller.



(a) Baseline. (b) Optimal.

Figure 5.8: Comparison of car side view for the baseline and optimal configurations. The figures show the roof drop of the optimal model. The underbody lift can not be seen from this angle.

5.3.5 Parameter effect

The optimal configuration of parameters lowered C_D substantially from the baseline configuration. But the parameters do not necessarily contribute equally, if at all, to that decrease. One way to measure the effect of each parameter is to compute the correlation coefficient (see section 3.7) between that parameter and C_D. The correlation coefficient measures the quality of a linear relation between each parameter and C_D. If the general trend is a decrease in C_D for an increased parameter value, this will result in a negative correlation coefficient. And vice versa; an increase in C_D will give a positive correlation coefficient. The parameter domain space was sampled with 10 points in every dimension, in total 100 000 data points. C_D was computed for all of these points using the neural network after which the correlation coefficients between the parameters and C_D were computed. Figure 5.9 shows the correlation coefficient for each parameter.

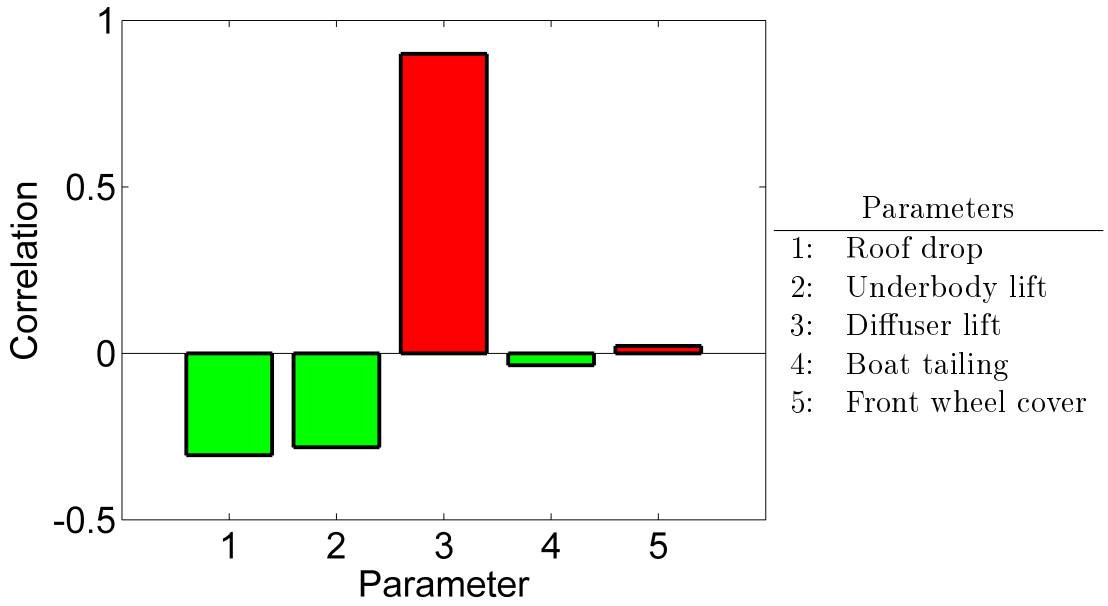


Figure 5.9: The bars show the correlation coefficient between each parameter and C_D. A positive correlation coefficient means C_D increases when the parameter increases and a negative correlation coefficient that C_D decreases.

As the figure illustrates the largest decrease in C_D comes from increasing roof drop and underbody lift. Boat tailing also lowers C_D but this effect is small compared to the effect of roof drop and underbody lift. Diffuser lift has the opposite effect with a large increase in C_D. The effect of increased front wheel cover is negligible.

It is however possible to have a decrease or increase in C_D at some intermediate parameter value but not at the end values. For such a non-linear relation the correlation coefficient is not a good measure. Another way to visualise the effect each parameter has on C_D is to plot C_D for all data points as a function of one parameter, e.g. roof drop. The average C_D is computed at each value of the roof drop to see if it changes when the roof drop changes.

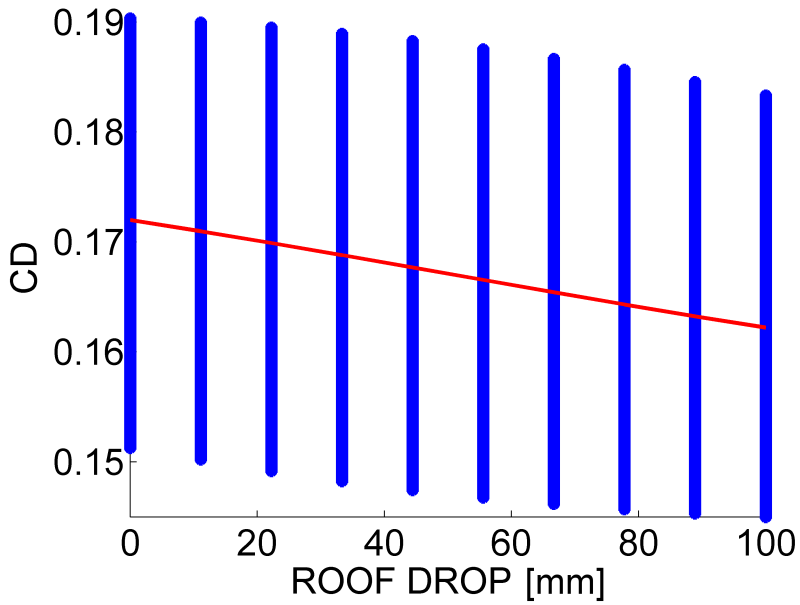


Figure 5.10: C_D of all points as a function of roof drop. The vertical blue lines are made up of C_D values for all 100 000 data points. The red line represents the average C_D at every roof drop value. There is a decrease in C_D for increasing roof drop.

.

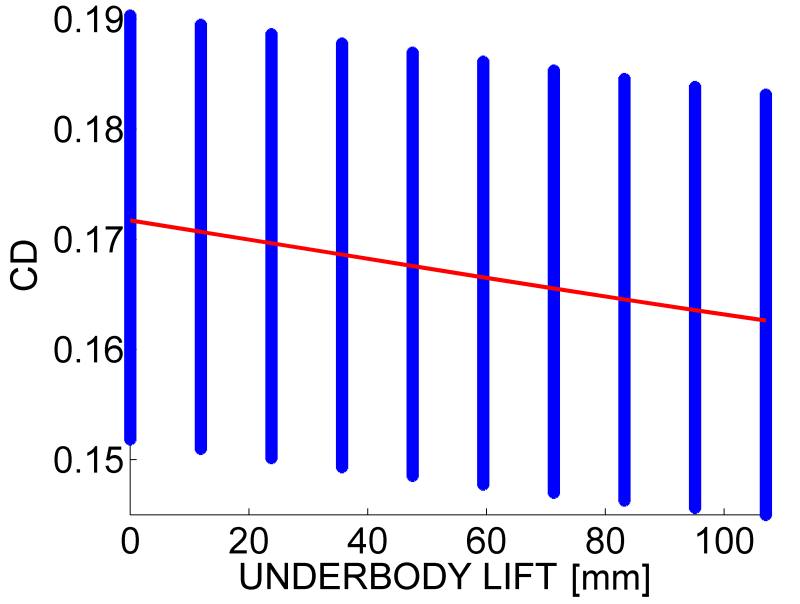


Figure 5.11: C_D of all points as a function of underbody lift. The vertical blue lines are made up of C_D values for all 100 000 data points. The red line represents the average C_D at every underbody lift value. There is a decrease in C_D for increasing underbody lift. The decrease is of the same magnitude as the effect of roof drop.

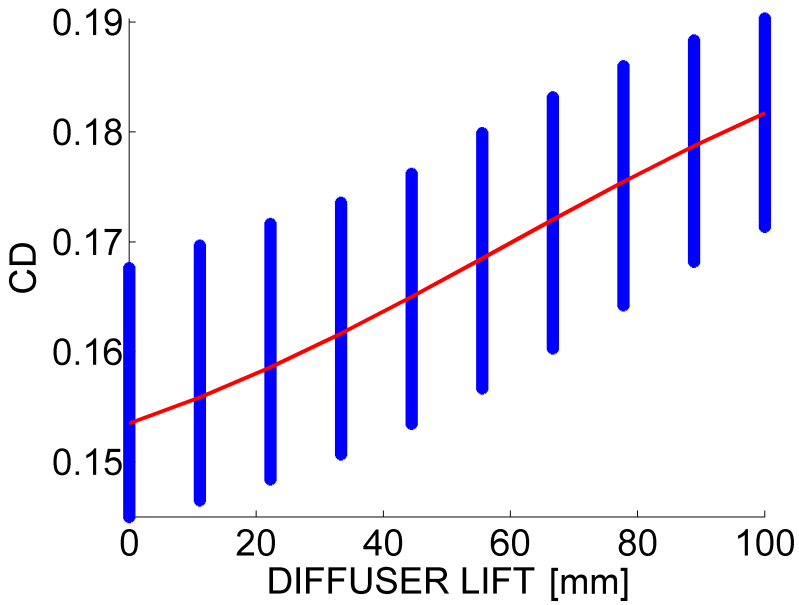


Figure 5.12: C_D of all points as a function of diffuser lift. The vertical blue lines are made up of C_D values for all 100 000 data points. The red line represents the average C_D at every diffuser lift value. There is a large increase in C_D for increasing diffuser lift.

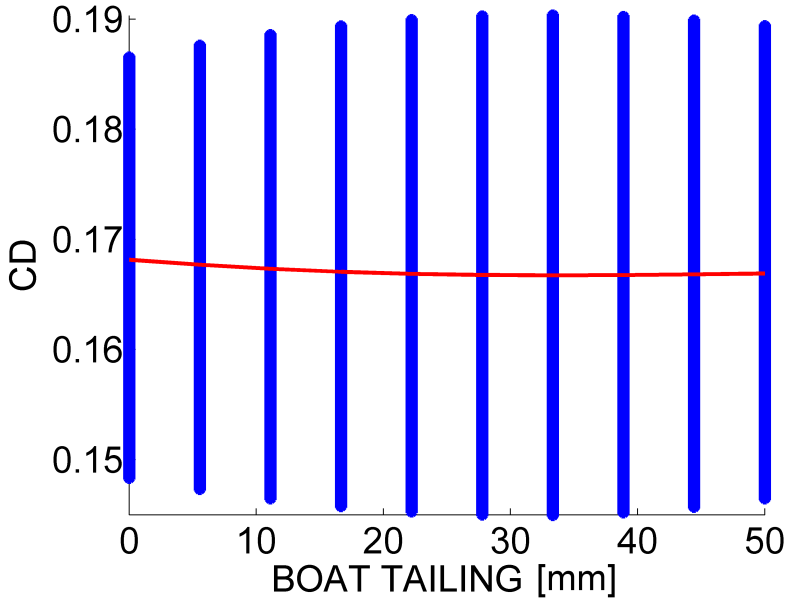


Figure 5.13: C_D of all points as a function of boat tailing. The vertical blue lines are made up of C_D values for all 100 000 data points. The red line represents the average C_D at every boat tailing value. There is a decrease in C_D at first, followed by an increase, for increasing boat tailing. The minimum C_D is found for a central value of the boat tailing.

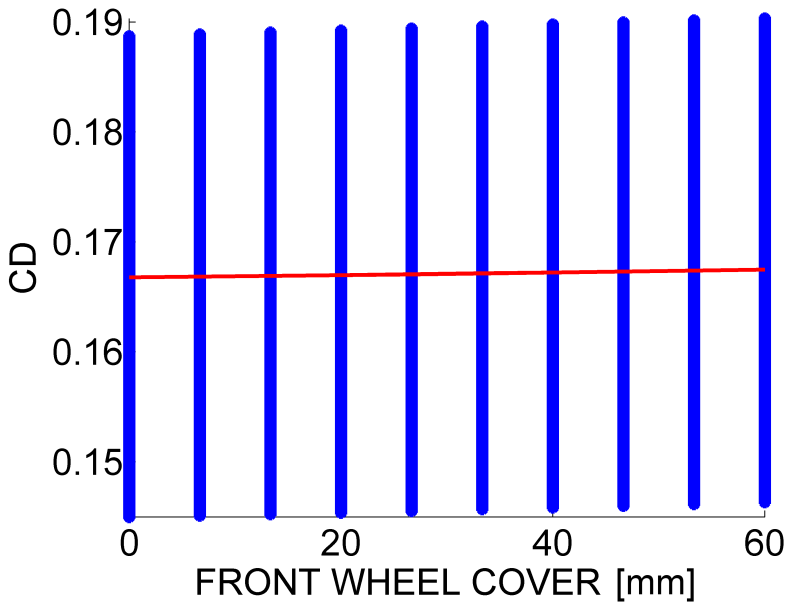


Figure 5.14: C_D of all points as a function of front wheel cover. The vertical blue lines are made up of C_D values for all 100 000 data points. The red line represents the average C_D at every front wheel cover value. There is no considerable change in C_D for increasing front wheel cover.

5.3.6 Time consumption

There are two aspects to time consumption; manual preparation time and computation time. The main part of the manual time is to set up the task in ANSA and prepare the morph boxes. Apart from that it is basically just running two scripts. How long the setup takes is very dependent on the complexity of the model and individual skills in ANSA. The morph boxes used in this study took around two days to build without prior experience of morphing in ANSA.

The computation time in generating the database is the most time consuming part of this process since it involves CFD simulations of a large number of cases. For this study of 34 simulations the computation time was around 20 hours on a computational cluster with 480 cores. The optimization task took less than 5 seconds on 2 cores.

5.4 Considering lift

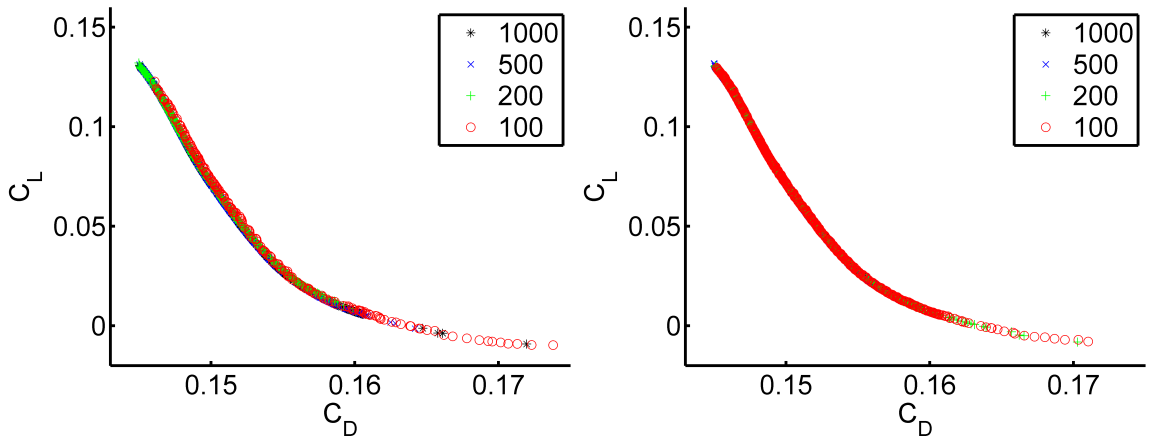
The extension of the model optimization study aimed to lower both C_D and C_L. It used the same database as the model optimization database.

5.4.1 Simulation convergence

As for C_D in the model optimization study the standard deviation of C_L was computed for all simulations. The maximum standard deviation of C_L is $7.8 \cdot 10^{-3}$ which corresponds to a 95% confidence interval with a deviation of 0.016 from the mean value. The average standard deviation is $1.5 \cdot 10^{-3}$, which is considerably larger than for C_D.

5.4.2 Optimization convergence

Since the multi-objective optimization does not converge to a single optimal point it is hard to define a convergence measurement similar to the one in Figure 5.5. It is however possible to check if the Pareto front is particle and generation independent, similar to the concepts of grid and iteration independence of the flow solver. Optimizations were performed for 50 and 100 particles with different number of generations, see Figure 5.15.



(a) 50 particles.

(b) 100 particles.

Figure 5.15: Pareto fronts for varying number of generations.

For 50 particles the Pareto front appears to be converged after 500 generations. With 100 particles the Pareto front is almost converged after 100 generations with only a few particles off from the main front. 200 generations seems to be enough for convergence. When comparing the solutions for 50 and 100 particles it can be seen that they have converged to the same Pareto front.

5.4.3 Convergence of neural network

Five simulations were run with 100 particles and 200 generations using different initialisations of the neural network. The resulting Pareto fronts are shown in Figure 5.16. Four simulations reached the same Pareto front, with one simulation reaching a separate front.

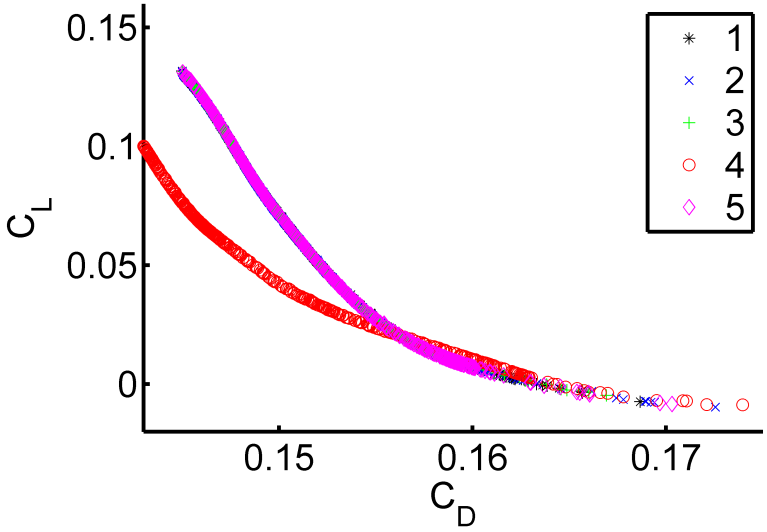


Figure 5.16: Five simulations with differently initialised neural networks.

5.4.4 Optimal configurations

The optimal attainable combinations of C_D and C_L are shown in Figure 5.17. Each point corresponds to a unique configuration of design parameters.

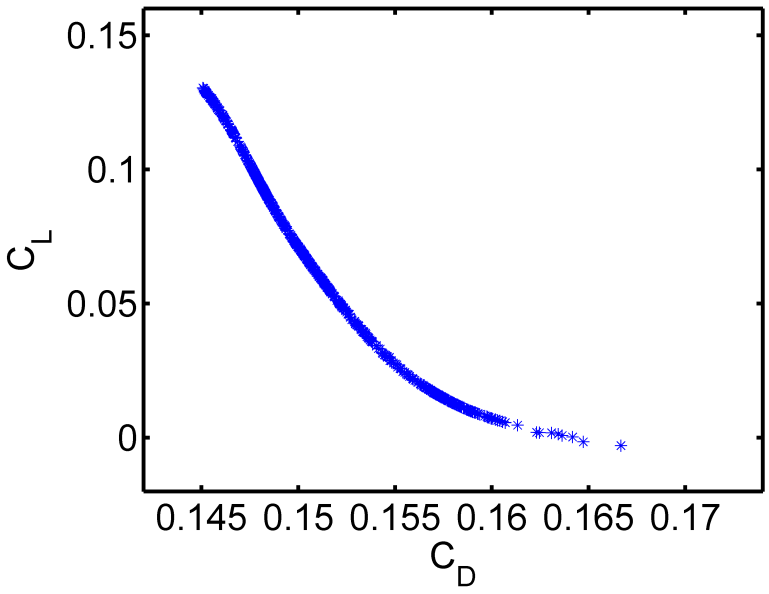


Figure 5.17: Optimal configurations of C_D and C_L.

It can be seen that C_D ranges from 0.145 to 0.167 and that C_L ranges from 0.00 to 0.13. Confirmation simulations were run for four points. These points were chosen as the optimal point for four different constraints on C_L; $C_L < 0.00$, $C_L < 0.04$, $C_L < 0.08$ and $C_L < 0.12$. The predicted points and their configurations are shown in Table 5.3.

Table 5.3: Four points on the predicted Pareto front.

Point	1	2	3	4
Constraint	$C_L < 0.00$	$C_L < 0.04$	$C_L < 0.08$	$C_L < 0.12$
C_D	0.165	0.153	0.149	0.146
C_L	-0.002	0.040	0.080	0.120
Roof drop [mm]	0.0	65.7	100.0	100.0
Underbody lift [mm]	1.5	4.6	38.6	88.3
Diffuser lift [mm]	13.6	0.0	0.0	0.0
Boat tailing [mm]	50.0	44.2	41.5	37.8
Front wheel cover [mm]	2.2	0.2	0.0	0.0

The predicted C_D and C_L values for these points are compared with simulation results in Table 5.4.

Table 5.4: Comparison of predicted and simulation values of C_D and C_L for four points on the predicted Pareto front.

	C_D		C_L	
	Predicted	Simulation	Predicted	Simulation
Point 1	0.165	0.174	-0.002	0.017
Point 2	0.153	0.157	0.040	0.045
Point 3	0.149	0.155	0.080	0.092
Point 4	0.146	0.146	0.120	0.141
Average error	0.005		0.014	

As seen in the table simulation results differ from the predicted values. The average error for the predicted C_D is 0.005 and the average error for the predicted C_L is 0.014.

5.4.5 Time consumption

The multi-objective optimization took slightly longer time than the optimization of only C_D. The computation time was less than 60 seconds on 2 cores.

5.5 Database size

The prediction error of the neural network was computed for five control points. The five control points were the same control points as for the Pareto front with the addition of the optimal configuration from the model optimization study. The errors are presented for different database sizes in Figure 5.18.

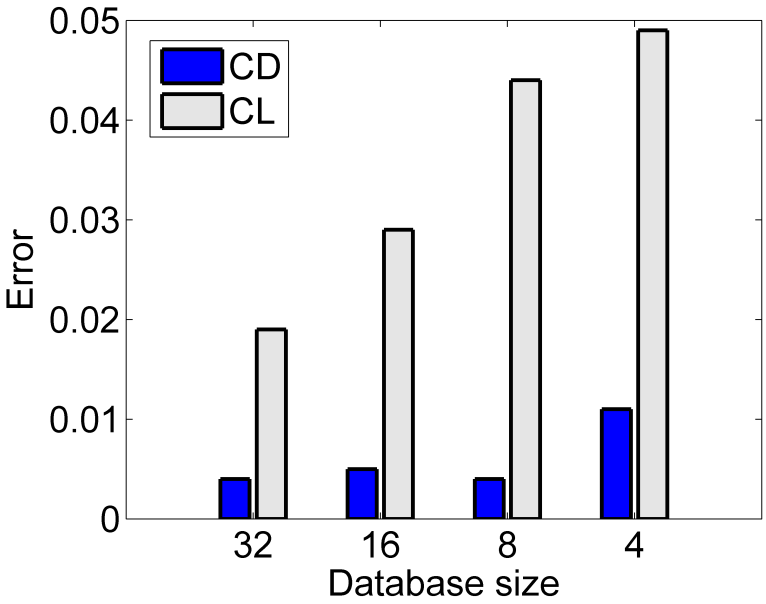


Figure 5.18: Average prediction error of neural network for different sizes of database.

6 Discussion

The following chapter contains a discussion of the obtained results.

6.1 Method validation

With the help of the developed optimization tool an improved solution was determined. The improved C_D was accurately predicted, indicating that the optimization tool performs well. The custom code *runOptimization* was found to be both more accurate and considerably faster than the commercial software modeFrontier for this specific case.

6.2 Convergence study

The best results were achieved with the under-relaxation factors of pressure and momentum set to 0.3. A study over 10000 iterations on a grid of 12.9 million cells converged after 2000 iterations with a variation in C_D from the mean value of less than one thousand. This precision is deemed good enough that the solution can be considered iteration independent.

The finer grid yielded a slightly lower C_D than the original grid. In fact the confidence intervals are separate, which means that the simulations are not grid independent. This is not ideal and one would preferably try to reach a grid independent solution. The focus of this thesis is however not to improve the CAE method but to implement an optimization tool and therefore the default grid size is left as it is. The chosen CAE method is a good trade-off between speed and accuracy as around 2000 iterations is sufficient to reach a 'good enough' solution. Refining the mesh will increase solution time. For the current optimization task where many simulations will be executed this is very time-consuming and expensive. Grid independence should also be of small importance for the optimization task since the same grid size is used for all cases and relative changes are the main interest.

6.3 Model optimization study

The optimization tool was proven to perform well. It was able to deliver a model with a 12.6% lower C_D than the baseline with a prediction error of just 0.3%. This was done by studying realistic changes of five design parameters on a car model that already had a low C_D. The entire process can be completed in three days.

The generated configurations have a larger minimum distance between any two configurations than a full-factorial sampling would, indicating a good coverage of the configuration space. The simulations of these configurations all converge well, enabling an accurate neural network.

It was found that 50 particles and 100 generations was more than enough to achieve convergence of the particle swarm optimization. The optimization was run several times with different values of particles and generations, each time for a newly initialised network. All runs yielded the same results, showing that the optimization method is robust.

The optimal configuration had a maximal roof drop of 100 mm, a maximal underbody lift of 107 mm and a boat tailing of 31.7 mm. There was no change in front wheel cover or diffuser lift. The design parameters contributed very differently to the change in C_D. Front wheel cover and boat tailing had almost no effect on C_D. This means that these can be chosen as any value in the domain without affecting the result, which can be of interest in a design process. The other three design parameters had an almost linear relation with C_D. Increased diffuser lift caused a large increase in C_D which is why the optimal configuration showed no change in diffuser lift. Roof drop and underbody lift caused a linear decline in C_D, with the effect of roof drop being slightly larger.

It is interesting to note that the three design parameters that caused most of the change in C_D (roof drop, underbody lift and diffuser lift) were either at their minimum or maximum values in the optimal configuration. This indicates that there is room for even further improvements by expanding the search domain. Increasing roof drop and underbody lift beyond their maximum values and moving the diffuser lift in the opposite direction will probably reduce C_D even further. However, expanding the domain limits will produce a less realistic car and might not even be possible. It is recommended that domain limits are set as the maximum realistic values from the start.

The optimal configuration lowered C_D from 0.167 to 0.146, a difference of 0.019 or 12.6%. This difference is computed from simulation results, meaning that it is independent of any uncertainty in the optimization prediction. The predicted C_D differs 0.3% from the simulation value. This one data point can not be taken as an indicative of the uncertainty in the optimization prediction. A better value is given in Table 5.4, where the average prediction error for four points on the Pareto front is presented. The average error in C_D is 0.005 for these points. Normalised with the average C_D this corresponds to around 3%. This is a relatively small error and the accuracy of the C_D prediction in the optimization tool is acceptable.

With the developed optimization tool an optimal configuration for a combination of five design parameters can be obtained in three days, which is a great improvement over

a classical approach of changing one parameter at a time. The main part of the time consumption is the creation of morph boxes and the solver computation time.

6.4 Considering lift

Considering both C_D and C_L in the optimization generates a Pareto front of optimal configurations. The Pareto front shows that optimizing for C_D will give an increase in C_L. The choice of a best configuration depends on the desired limits on C_D and C_L and will vary for each application.

Convergence was reached using both 50 and 100 particles, but 50 particles required more generations. A converged solution is obtained at the lowest cost for 100 particles and 200 generations. Using 100 particles also seems to give a better spread of the Pareto front than 50 particles.

The simulated results for the four points on the predicted Pareto front show the same trend as the predicted values, but the absolute values differ. The values of both C_D and C_L are underpredicted by the optimization tool. The average prediction error of C_D is 0.005, which is a relatively small error. The prediction error of C_L is worse at 0.014. This difference is probably due to the difference in convergence of C_D and C_L. The average standard deviation of C_D in the simulations was almost an order of magnitude smaller than the average standard deviation of C_L. This larger uncertainty in the training database creates a less accurate neural network and hence a larger uncertainty in the prediction.

The optimization tool does not reach the same Pareto front for all runs, even if most do. This means that the convergence of the neural networks is not entirely consistent. This problem did not occur for the optimization of C_D where all runs gave the same result, indicating that the problem might be located in the neural network for C_L, possibly an effect of the higher solver uncertainty of C_L.

6.5 Database size

As expected the best accuracy is achieved with the largest database. The prediction error of C_L grows steadily with a decreasing database size. The prediction error of C_D is however quite constant for a database size of 8 to 32 elements, but increases when the database size is lowered to 4 elements.

Which database size is required depends on the application and the desired accuracy, but also on the convergence of the underlying solver. Lowering the prediction error can thus be done in two ways; improving the solver or increasing the database size. Improving the solver can be done by for example changing the turbulence model. A more accurate solver is however likely to be more computationally expensive - as always there is a trade-off between accuracy and speed. Increasing the database size will also increase computation

time. Which of these approaches is the most efficient is unclear and such an investigation is outside the scope of this thesis.

7 Conclusion

A fast method for optimization of a vehicle's shape was developed and an optimization tool compatible with Volvo's CAE process was delivered. With the help of the optimization tool it is possible to determine the optimal configurations of a vehicle's shape. Drag and lift coefficients are predicted with an acceptable accuracy. The optimization tool requires the construction of morphing boxes as the only manual work, the rest is automated. It enables a study of several design parameters to be carried out in a few days, which is a great improvement over a classical approach of changing one parameter at a time.

The optimization tool was used on a simplified low-drag car model in a study of realistic changes of five design parameters. It was able to deliver an improved shape, with a 12.6% lower C_D. The prediction error of C_D was 0.3%.

7.1 Future work

The proposed optimization tool performs well, but has only been tested for one solver and with a limited database. There is a potential for improving the accuracy of the tool by improving the convergence of the solver and increasing the database. An interesting extension of the presented work would be a detailed study of how the accuracy depends on the quality and size of the database. Such a study would aim to find the best trade-off between speed and accuracy for the optimization tool.

Another area of interest for future work is to reduce the time spent on constructing morph boxes, which is the most time-consuming manual part of the optimization tool. Simplifying and automating this process has a potential of greatly reducing the manual work required.

Bibliography

Fasihul M. Alam, Ken R. McNaught, and Trevor J. Ringrose. A comparison of experimental designs in the development of a neural network simulation metamodel. *Simulation Modelling Practice and Theory*, 12(7–8):559 – 578, 2004. ISSN 1569-190X. doi: 10.1016/j.simpat.2003.10.006.

R.H. Barnard. *Road Vehicle Aerodynamic Design*. MechAero Publishing, St Albans, 3rd edition, 2009.

Mark Hudson Beale, Martin T. Hagan, and Howard B. Demuth. Matlab neural network toolbox: User's guide. http://www.mathworks.com/help/pdf_doc/nnet/nnet_ug.pdf, March 2014. Retrieved: 2014-03-13.

Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, June 2004. doi: 10.1109/TEVC.2004.826067.

Lars Davidson. An introduction to turbulence models. Technical Report 97/2, Department of Thermo and Fluid Dynamics, Chalmers University of Technology, 2011.

Lars Davidson. Fluid mechanics, turbulent flow and turbulence modeling. Department of Applied Mechanics, Chalmers University of Technology, March 2013. Lecture Notes in Turbulence Modeling.

Laurent Dumas. Cfd-based optimization for automotive aerodynamics. In D. Thévenin and G. Janiga, editors, *Optimization and Computational Fluid Dynamics*, chapter 7. Springer-Verlag, Berlin, 2008. doi: 10.1007/978-3-540-72153-6.

Emad Elbeltagi, Tarek Hegazy, and Donald Grierson. Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, 19 (1):43 – 53, 2005. ISSN 1474-0346. doi: 10.1016/j.aei.2005.01.004.

Kai-Tai Fang, Runze Li, and Agus Sudjianto. *Design and Modeling for Computer Experiments*. Chapman and Hall/CRC, Boca Raton, 2006. doi: 10.1201/9781420034899.

F. Dan Foresee and Martin T. Hagan. Gauss-newton approximation to bayesian learning. In *International Conference on Neural Networks*, volume 3, pages 1930–1935, 1997.

Kyriakos C. Giannakoglou and Dimitrios I. Papadimitriou. Adjoint methods for shape optimization. In D. Thévenin and G. Janiga, editors, *Optimization and Compu-*

tational Fluid Dynamics, chapter 4. Springer-Verlag, Berlin, 2008. doi: 10.1007/978-3-540-72153-6.

David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison Wesley, 1989. ISBN 0-201-15767-5.

Simon Haykin. *Neural Networks and Learning Machines*. Pearson Education, Upper Saddle River, 3rd edition, 2009. ISBN 978-0-13-129376-2.

Eysteinn Helgason and Haukur Elvar Hafsteinsson. Automatic shape optimization of aerodynamic properties of cars. Master's thesis, Chalmers University of Technology, 2009.

IPCC. Climate change 2013: The physical science basis, headline statements. https://ipcc.ch/news_and_events/docs/ar5/ar5_wg1_headlines.pdf, September 2013. Retrieved: 2014-02-10.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks*, volume 4, pages 1942–1948. Perth, Australia, 1995.

Jong-Eun Kim, Vinay N. Rao, Roy P. Koomullil, Doug H. Ross, Bharat K. Soni, and Alan M. Shih. Development of an efficient aerodynamic shape optimization framework. *Mathematics and Computers in Simulation*, 79(8):2373 – 2384, 2009. ISSN 0378-4754. doi: 10.1016/j.matcom.2009.01.012.

Robert Louis Lietz. Vehicle aerodynamic shape optimization. Technical Report 2011-01-0169, SAE, 2011. doi: 10.4271/2011-01-0169.

Frédérique Muyl, Laurent Dumas, and Vincent Herbert. Hybrid method for aerodynamic shape optimization in automotive industry. *Computers and Fluids*, 33(5–6):849 – 858, 2004. ISSN 0045-7930. doi: 10.1016/j.compfluid.2003.06.007.

European Commission. Reducing co2 emissions from passenger cars. http://ec.europa.eu/clima/policies/transport/vehicles/cars/index_en.htm, January 2014a. Retrieved: 2014-02-10.

European Commission. The 2020 climate and energy package. http://ec.europa.eu/clima/policies/package/index_en.htm, January 2014b. Retrieved: 2014-02-10.

European Environment Agency. Total final energy consumption by sector in the eu-27, 1990-2010. http://www.eea.europa.eu/data-and-maps/figures/final-energy-consumption-by-sector-6, December 2012. Retrieved: 2014-02-10.

John A. Rice. *Mathematical Statistics and Data Analysis*. Thomson Brooks/Cole, Belmont, 3rd edition, 2007. ISBN 0-495-11868-0.

Tsan-Hsing Shih, William W. Liou, Aamir Shabbir, Zhigang Yang, and Jiang Zhu. A new $k-\varepsilon$ eddy viscosity model for high reynolds number turbulent flows. *Computers and Fluids*, 24(3):227–238, 1995. doi: 10.1016/0045-7930(94)00032-T.

S.N. Singh, L. Rai, P. Puri, and A. Bhatnagar. Effect of moving surface on the aerodynamic drag of road vehicles. In *Proceedings of the Institution of Mechanical Engineering*, volume 219, pages 127–134. Journal of Automobile Engineering, 2005.

K.S. Song, S.O. Kang, S.O. Jun, H.I. Park, J.D. Kee, K.H. Kim, and D.H. Lee. Aerodynamic design optimization of rear body shapes of a sedan for drag reduction. *International Journal of Automotive Technology*, 13(6):905–914, 2012. ISSN 1229-9138. doi: 10.1007/s12239-012-0091-7.

René A. Van den Braembussche. Numerical optimization for advanced turbomachinery design. In D. Thévenin and G. Janiga, editors, *Optimization and Computational Fluid Dynamics*, chapter 6. Springer-Verlag, Berlin, 2008. doi: 10.1007/978-3-540-72153-6.

Richard M. Wood. Impact of advanced aerodynamic technology on transportation energy consumption. Technical Report 2004-01-1306, SAE, 2004. doi: 10.4271/2004-01-1306.