Individual Report – I2

ECMM102

Title: Virtual Wind Tunnel

Date of submission: 01/05/2014

Student Name: Heather Bolt
Programme: MEng Mechanical Engineering
Student number: 610033357
Candidate number: 022518

Supervisor: Dr. Gavin Tabor

Abstract

This individual project is part of the Virtual Wind Tunnel group project, which aimed to , to investigate the integrated design approach to design the external geometry of a concept car using: Computer Aided Design (CAD), CFD, and experimental testing. This report details the individual work undertaken by the author for this group project.

A virtual wind tunnel is one that has been created using CFD. Due to advances in computational power, CFD has been increasing in popularity in the automotive industry. However, CFD simulations can be very computationally expensive and take days or even weeks to run. Therefore in many cases, CFD-based optimisation studies are not feasible. The aim of this project was to investigate the use of a surrogate-based modelling technique to predict the drag coefficient of a simplified car body (the Ahmed body). A CFD model of the flow around the Ahmed body was created and validated against experimental data. An automated process was then developed to run multiple CFD simulations of the flow around the Ahmed body - varying the rear slant angle and a fillet radius of the sharp edges. A surrogate model was created to predict the drag coefficient of the Ahmed body, given the two geometric input parameters. The surrogate model was significantly computationally cheaper than the CFD model. It was also found to be in good agreement with the CFD simulations and was able to predict the drag coefficient for the Ahmed body with an average error of 2.3%.

Keywords:

CFD, OpenFOAM, Ahmed Body, Surrogate Modelling

Acknowledgements

Special thanks to Dr. Gavin Tabor for his supervision of this project and his expert guidance with OpenFOAM. I would also like to acknowledge Prof. Richard Everson for his assistance in the machine learning aspect of this project.

Table of Contents

1 Introduction

A key tool utilised in automotive vehicle development is the wind tunnel - a device that is used to study the flow of air past an object. With recent advances in computational power, the virtual wind tunnel (VWT) has been gaining popularity in the automotive industry for analysis, design and optimisation. A VWT is one that has been created using Computational Fluid Dynamics (CFD), a method that involves using a computer to solve fluid flows. Its rise in popularity is caused by a number of factors, namely: it allows the user to have complete control of the virtual environment; it is a powerful visualisation tool that enables better understanding of complex flow phenomena; a vast amount of information can be yielded at any point within the virtual space; and physical modifications are not necessary, making it quicker and cheaper than a physical wind tunnel.

There were three main objectives for the VWT group project. Firstly, to investigate the integrated design approach to design the external geometry of a concept car using: Computer Aided Design (CAD), CFD, and experimental testing. Secondly, to explore the use of high performance computing (HPC) to create and validate a VWT. Thirdly, to redesign the University of Exeter's current wind tunnel facilities and use them to obtain experimental results to compare with the VWT. To achieve these objectives the group was divided into three subgroups: automotive design, experimental design and HPC.

There were two main objectives for this project (which was part of the HPC sub-group). Firstly, to create a CFD model of the flow around a simplified car body (the Ahmed body) that could produce the quality and quantity of data required to train a surrogate model. Secondly, to create a surrogate model for the Ahmed body to investigate the use of this modelling technique within the automotive design process. The purpose of a surrogate model is to mimic the input/output behaviour of a more complex computational model, but at a much lower computational expense. A single CFD calculation can take days to run and so a surrogate model allows the user to run hundreds (or even thousands) of simulations for design optimisation where it would have been impossible otherwise. The project deliverables were as follows:

- Learn OpenFOAM [1] (an open source CFD software package) to enable the creation of a VWT using CFD.
- Evaluate and choose appropriate boundary conditions for the CFD wind tunnel, such as: inlet and outlet parameters, size of domain and turbulence model.
- Validate the Ahmed body VWT with experimental data.
- Automate: the creation of model geometry; the meshing process; and the solution output. Use this automated process to run multiple simulations to provide data to train a surrogate model.
- Create a surrogate model for an Ahmed body to investigate the use of a surrogate-based modelling approach in automotive design.

Firstly, this report will summarise the background for this project and give a review of relevant literature. Next, an outline of the project methodology will be given. Following this, the design of the virtual wind tunnel for the Ahmed body is described before the results are presented and analysed. Afterwards, the design of the surrogate modelling process is explained and the results are presented and analysed. Finally, sustainability is discussed and project conclusions are drawn.

2 Background & Review of Previous Work

This section of the report provides a summary of the information that was previously discussed in the background section of *Individual Report - I1* [2], in addition to presenting new information.

2.1 The Ahmed Body

The Ahmed body is a simplified model of the geometry of a generic car - the dimensions of which can be seen in Figure 2.1 [3] below. This model has a rounded front and a slanted back, which has an angle (α) that can be varied. It was first defined by Ahmed et al. in the experimental study - *Some Salient Features Of The Time-Averaged Ground Vehicle Wake* [4]. Despite being a relatively simple model, 'the flow around it retains some main features of the flow around real cars' [5]. Thus it is often used as a benchmark case for investigating automobile flow. Due to the amount of experimental and computational literature available on the Ahmed body, it was decided by the HPC sub-group to use the Ahmed body to develop the virtual wind tunnel.

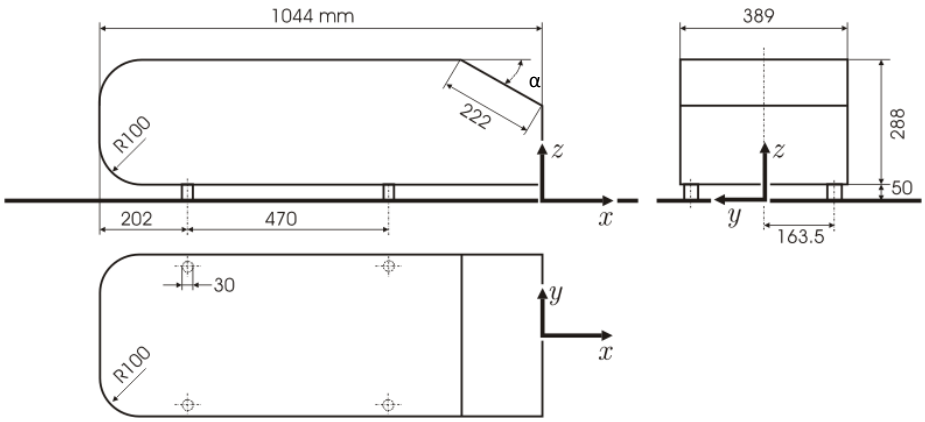


Figure 2.1 - Ahmed Body [3]

2.1.1 Drag Force

There are two types of drag force acting on the Ahmed body: skin friction drag and pressure drag (also called form drag). Skin friction drag is dependent on 'the shearing forces acting between the body and the fluid', whereas pressure drag depends on 'the pressure forces acting on the body' [6]. The initial study by Ahmed et al. on this model revealed that 85% of the drag on the Ahmed body was pressure drag, and most of this drag was generated at the rear end of the model [4]. To quantify the drag acting on the Ahmed body the dimensionless drag coefficient is used [6]:

Equation 1

$$C_D = \frac{2F}{AU_0^2\rho}$$

Where: C_D – drag coefficient; F – drag force (N); A – projected area (m^2); U_0 – free stream velocity (m/s); ρ – density (kg/m^3)

2.1.2 Rear Slant Angle

Through experimental investigations, Ahmed et al. demonstrated that the drag on the Ahmed body is dependent on the rear slant angle (α). Figure 2.2 [7] shows the relationship between the drag coefficient and the rear slant angle. This figure shows that there are two critical angles for the flow around the Ahmed body: one at α = 12.5° (point B) where the drag is at a minimum, and one at α = 30° (point C) where the drag is at a maximum. When the rear slant angle is increased between 0° and 12.5° the flow remains fully attached on the rear slant. This attachment is caused by two counter rotating vortices, which form at the upper corners of the slant, as the rear slant angle is increased from 0°. These vortices generate downwash that 'promotes attached flow on the central portion of the slant surface' [7]. Between 12.5° and 30° the flow separates and reattaches on the rear slant surface. Increasing the rear slant angle increases the strength of the counter rotating vortices thus producing an increase in drag. At α > 30° the trailing vortices lift off and/ or burst [7] and the flow abruptly detaches from the back slant - causing a sudden reduction in the drag on the body. See Figure 2.3 [8] for a sketch of the flow in the wake region of the Ahmed body as a function of α.

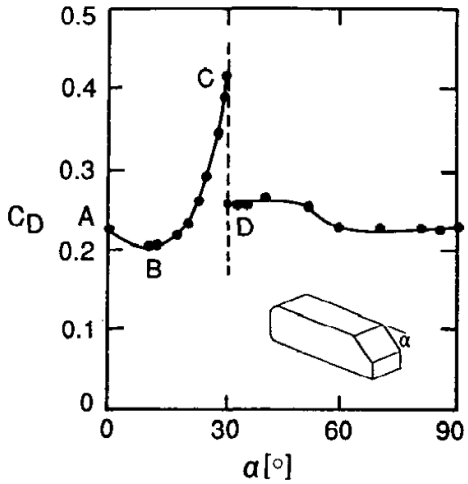


Figure 2.2 – Dependence of drag coefficient C_D on the rear slant angle α [7]

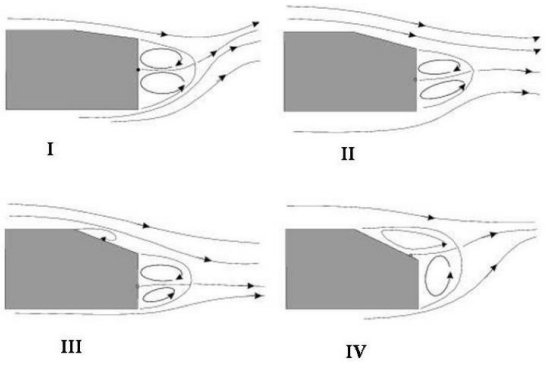


Figure 2.3 – Sketch of flow in the wake region. I & II for α < 12.5°; III for 12.5 < α < 30°; IV for α > 30° [8]

2.1.3 The Ahmed Body in Literature

Since the original experiments on the Ahmed body by Ahmed et al., there have been several other experimental investigations [9, 10, 11, 12]. Most notably, in 2003 Lienhart et al. performed a detailed experimental analysis on the Ahmed body at α = 25° and 35° [13] (either side of the 30° critical slant angle). The purpose of this study was to provide 'quantitative information . . . for developing, testing, and validating computer models of the aerodynamics of vehicular wake regions' [13]. The results from this experiment are freely available online, therefore the majority of the literature on CFD studies of the Ahmed body is based on this experiment. In addition, the European Research Community On Flow, Turbulence And Combustion (ERCOFTAC) held a workshop on refined turbulence modelling [14] where one of the test cases was the Ahmed body. Participants were asked to model the Ahmed body and compare their results to those of Lienhart et.al. For this workshop, ERCOFTAC gave several recommendations for the computational modelling of the Ahmed body and these guidelines have been adopted in similar CFD

investigations that were not part of the original workshop [11,15]. For this project, it was decided to replicate the experiments performed by Lienhart et al. using CFD and utilise the guidelines recommended by ERCOFTAC.

In the experimental study by Lienhart et al. the free stream velocity was set at 40 m/s, which corresponds to a Reynolds number of 2.78×10^6 (based on the Ahmed body model length of 1.044m). On the other hand, in the experiments by Ahmed et al. the free stream velocity was 60 m/s, which gives a Reynolds number of 4.18×10^6. This difference in Reynolds number has little effect on the general flow patterns around the Ahmed body as 'the external vehicle flow at high Reynolds number becomes insensitive to the Reynolds number' [16]; instead it is 'the geometry [that] dictates the character of the flow (attached or detached) and the position of flow separations' [16]. However, as the drag coefficient is a function of the Reynolds number [17] the drag coefficient at $R_e = 2.78\times10^6$ is not the same as $R_e = 4.18\times10^6$. Therefore in this report the values of the drag coefficients at different rear slant angles will not be the same as those measured by Ahmed et al. In their experiments Lienhart et al. do not measure the drag coefficient, however it has been experimentally measured at different Reynolds numbers in several studies [9, 11]. For this project, drag coefficient data will be taken from a report by Meile et al. [11], who performed experimental and computational analysis on the Ahmed body under similar conditions to Lienhart et al.

One notable absence in the literature on Ahmed body CFD models, is a range of rear slant angles. In all of the literature that the author has read for this project, the CFD models only used one or two rear slant angles. However, in the experiments by Ahmed et al. [4] the rear slant angle was varied from 0° to 40°, in 5° increments. It was therefore decided to use a range of rear slant angles for the Ahmed body, in order to assess the overall performance of the CFD wind tunnel, rather than its specific performance at one particular angle.

2.2 Computational Fluid Dynamics

2.2.1 Overview of Computational Fluid Dynamics

For this project, the virtual wind tunnel was created using CFD - a method that involves using a computer to solve fluid flows. The behaviour of fluids is governed by the Navier-Stokes equations - these equations describe the motion of the fluid in terms of its velocity, pressure, temperature and density. The Navier-Stokes equations are a set of coupled 2^{nd} order partial differential equations. There are five equations: a continuity equation for the conservation of mass, *x*-, *y*- and *z*-momentum equations and an energy equation [18]. The Navier-Stokes equations are too complicated to solve analytically without being simplified and so CFD uses a computer to solve approximations of these equations using a variety of methods. Central to most CFD codes is the finite volume method [19] (including OpenFOAM [20] - which was used in this project) and so this method will be briefly discussed here. The finite volume method is centered on dividing the fluid domain into a finite number of control volumes (known as cells or elements) - this is called a mesh. The numerical algorithm used to solve the Navier-Stokes equations consists of three steps [19]:

1. Integrating the Navier-Stokes equations over all the elements in the mesh to create a set of ordinary differential equations for each element.
2. Discretising the ordinal differential equations to produce a set of non-linear algebraic equations.

3. Linearising and solving the algebraic equations iteratively.

The principles of the finite volume method and the derivation Navier-Stokes equations will not be discussed here, as this theory is widely available in several textbooks (such as *An introduction to computational fluid dynamics: the finite volume method* by H.K. Versteeg, and W. Malalalsekera [19]).

2.2.2 OpenFOAM

The CFD code used in this project was OpenFOAM (Open source Field Operation And Manipulations) [1]. OpenFOAM is an open source '*C ++ library*, used primarily to create executables known as *applications*' [21]. Applications are classed as either solvers or utilities. Utilities are 'designed to perform tasks that involve data manipulation'. For example, the OpenFOAM utility `extrude2DMesh` takes a 2D mesh and extrudes it into a 3D mesh. Solvers are 'designed to solve a specific problem in continuum mechanics' [21]. For example, the OpenFOAM solver `simpleFoam` solves for incompressible, steady-state flow. For clarity, in this report all OpenFOAM code will be formatted as '`simpleFoam`' and all OpenFOAM directories and files will be formatted as '*`constant`*' (for example). There are three main steps involved in using OpenFOAM: pre-processing, solving and post-processing. The general structure of OpenFOAM can be seen in Figure 2.4 [21] below. During pre-processing: the geometry is defined; the fluid domain is meshed; and the boundary conditions (fluid behaviour and properties at the boundaries) are assigned. In the solving step the fluid problem is solved iteratively using the finite volume method. Finally, the results are visualised and analysed in the post-processing stage.

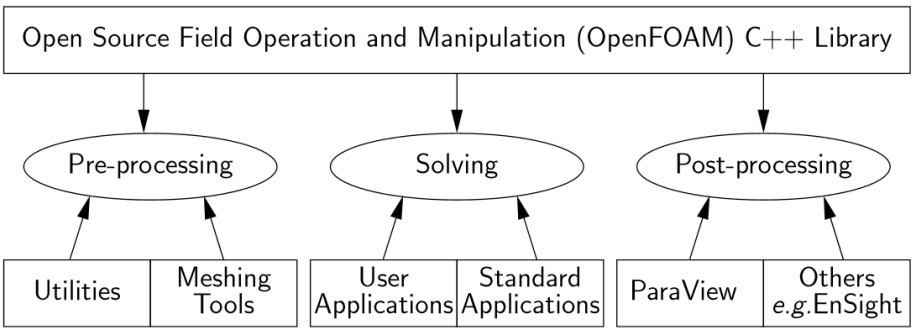


Figure 2.4 – General structure of OpenFOAM [21]

2.2.3 Turbulence Models

The flow in the wake region of the Ahmed Body is 'characterised by highly turbulent and three-dimensional separations' [15]. Hence the key to successfully predicting the drag on the Ahmed body is to correctly model the separation of flow on the car and the flow behaviour in the wake region. In contrast to the orderly motion of laminar flow, turbulent flow is characterised by the chaotic, random motion of particles. This makes turbulent flow very difficult to predict. Consequently, there is not one single turbulent modelling method. Instead there are numerous different approaches to modeling turbulent flow - choosing the most appropriate turbulence model is one of the main challenges of using CFD. In the literature there are three main types of turbulence model that have been used for the Ahmed body: RANS (Reynolds Averaged Navier Stokes), LES (Large Eddy Simulation) and DES (Detached Eddy Simulation). The theory behind these models is

widely available and so will not be discussed here. If the reader wishes to know more, then Chapter 3 'Turbulence and its modelling' in *An introduction to computational fluid dynamics: the finite volume method* by H.K. Versteeg, and W. Malalalsekera gives a detailed description of the theory behind the turbulence models discussed in this report.

In turbulent flow there is a spectrum of eddy sizes, and turbulence models are classified according to which turbulent scales they model and which turbulent scales they resolve. Figure 2.5 (below) is a schematic of the turbulent energy spectrum showing the different turbulence model approaches. RANS turbulence modelling uses time averaged Navier Stokes equations to estimate the effect that turbulence has on the mean flow. Consequently, the behaviour of all of the different sized eddies is described in one turbulence model. A different approach is LES turbulence modelling, which works on the basis that small scale eddies are statistically isotropic - whereas large scale eddies are determined by the geometry of the flow domain. Hence in LES turbulence modelling, the small scale eddies are modeled whereas the larger eddies are explicitly calculated with a time-dependent simulation. The third type of turbulence model that has been applied to the flow around the Ahmed body is a DES model. DES based turbulence models are effectively a hybrid between RANS and LES. These models use a RANS-type model at the near wall regions and a LES-type model in the free stream.

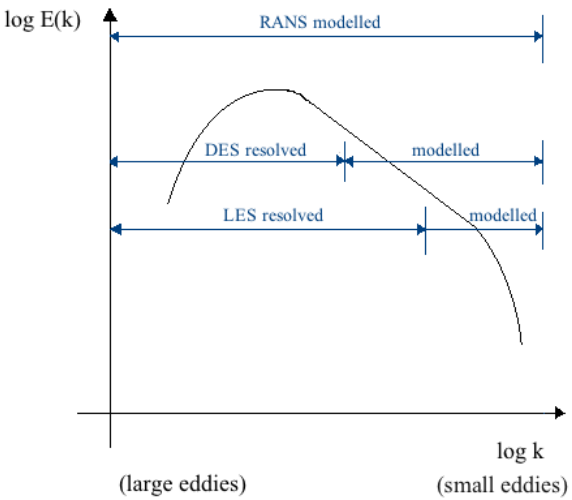


Figure 2.5 – Turbulent energy spectrum showing regions that are modeled or resolved by different turbulence models

There have been studies on the Ahmed body using LES turbulence models [5, 8, 22, 23, 24, 25], DES turbulence models [25, 26] and RANS turbulence models [11, 24, 26, 27]. As expected, the LES turbulence models perform better than DES [25] or RANS [24]. They are more accurate than the other two models and are therefore better at predicting the complex flow that occurs on the rear slant surface and in the wake region of the Ahmed body. However, the extra accuracy provided by LES turbulence models also comes at a higher computational cost. One study of LES and RANS turbulence modelling of the Ahmed body, recorded that the LES simulation took six days whereas the RANS simulations took 20 hours (when running 12 million cell meshes in parallel on 8 CPUs) [24]. For this project, one of the deliverables was to run multiple CFD simulations to provide a sufficient amount of data to train a surrogate model. Therefore it was decided that it would not be feasible to use DES or LES turbulence models due to the

computational time required to run these models (and so RANS turbulence models were used). However, Luke Hamilton's contribution to the group project involved investigating more accurate turbulence models for the Ahmed body.

2.2.4 Meshing

One crucial aspect of CFD modeling is the mesh - the division of the fluid domain into a finite number of control volumes (or elements). A fine mesh (a mesh with more elements) will give more accurate results than a coarse mesh, however the solution will also take longer to run. The meshing process is therefore a compromise between computational resources available and accuracy required; 'to have a suitable mesh density for a specific problem with suitable CPU time and numerical accuracy has remained a major theme in CFD literature' [27]. A wide range of mesh densities was found in the literature review - from 0.9 million [23] to 40 million [25] elements. The mesh density required to produce a satisfactory solution depends on a variety of factors. Most significantly: the turbulence model used (for example LES turbulence modelling requires a finer mesh at the wall boundaries), and the complexity of flow (at smaller slant angles before separation the flow is simpler to model). It is not only the quantity of elements that affect the accuracy of the final solution but also the quality. The quality of a mesh depends on various criteria such as 'the shape of the cells (aspect ratio, skewness, warp angle or included angle of adjacent faces), distances of cell faces from boundaries, and spatial distribution of cell sizes' [28].

The mesh in the near wall region requires careful consideration; as for the Ahmed body 'it is the main location of turbulence modelling' [16]. The near wall mesh is often quantified in terms of its y^+ value. This is the non-dimensional distance from the wall to the first node in the wall adjacent mesh and is defined as [29]:

Equation 2

$$y^+ = \frac{y}{\nu}\sqrt{\frac{\tau_w}{\rho}}$$

Where: y^+ - non-dimensional distance from wall to first node in the mesh; y – distance from wall to first node in the mesh (m); ν – local kinematic viscosity of fluid (m^2/s); τ_w – wall shear stress (Pa); ρ – fluid density (kg/m^3)

Figure 2.6 [30] highlights the different areas of flow in the near-wall region (known as the boundary layer flow). There are two ways to model the flow in the boundary layer. Firstly, by resolving the flow down the laminar sublayer of the boundary layer. To do so the y^+ value of the mesh must lie within the laminar sublayer region where $y^+ < 5$, and preferably should be around $y^+ \leq 1$, thus requiring a very fine mesh at the wall. The second method involves resolving the flow down to the log law region of the boundary layer, and using wall functions to bridge the results between the flow properties at the wall and the flow properties in near wall cells. Using wall functions requires the wall-adjacent cell centre in the mesh to lie within the log-law layer, which is in the range of $20 < y^+ < 200$ [31]. Figure 2.7 [32] shows the near-wall velocity profile for the flow in the boundary layer, and highlights the wall resolved approach (in blue) and the wall function approach (in red). In this project wall functions were used because, due to the computational expense, it would not be feasible to use a wall resolved approach.

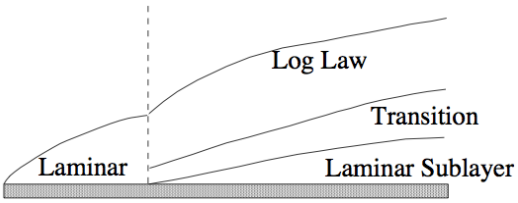


Figure 2.6 – Boundary layer flow [30]

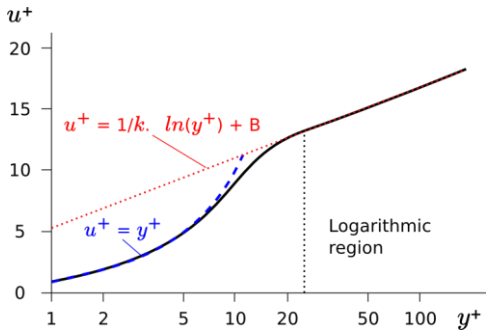


Figure 2.7 – Near-wall velocity profile for boundary layer flow [32]

To ensure the correct y^+ distance and to have a good quality mesh in the near wall region, it is common practice to use wedge elements next to the wall (known as a boundary layer mesh) and tetrahedral elements in the free stream [33]. An example of a tet-wedge mesh for an Ahmed body can be seen in Figure 2.8 [34].

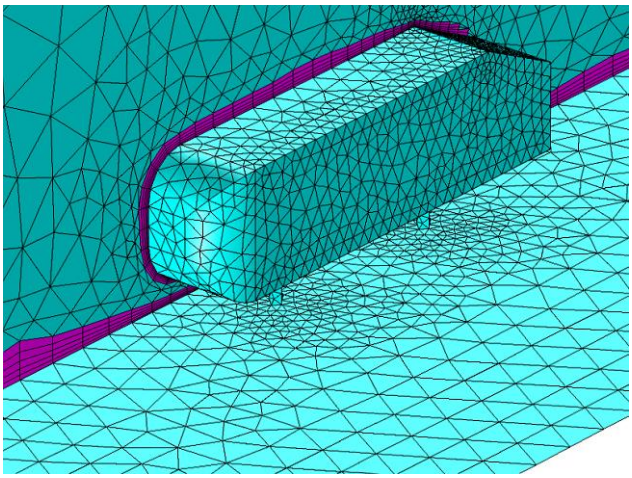


Figure 2.8 – Mesh around an Ahmed body, showing the boundary layer mesh (magenta) and the surrounding tetrahedral mesh (cyan) [34]

2.3 Surrogate Modelling

CFD simulations can be very computationally expensive and take days or even weeks to run. Therefore in many causes 'direct optimisation, which requires many calls to the model of interest, is more often than not, unrealistic' [35]. In these cases, a surrogate-based modelling approach may be used. Surrogate models are 'compact scalable analytic models that approximate the multivariate input/output behavior of complex systems, based on a limited set of computational expensive simulations' [36]. A surrogate model is not as accurate as the individual computational simulations that it is approximating, however it allows results to be gained within minutes rather than days. Thus allowing hundreds of results to be assessed where it would have been impossible otherwise. A surrogate-based modelling approach has successfully been used in CFD applications, including: the optimisation of labyrinth seals [37]; to compute vapor-water two-phase flows [38]; and to predict the temperature distribution in a semi-conductor chip [39]. In this project, a

surrogate model was created for the flow around the Ahmed body to investigate the use of a surrogate-based modelling approach in design optimisation.

2.3.1 Overview of Artificial Neural Networks

One common type of surrogate model is an Artificial Neural Network (ANN), which was used in this project. Figure 2.9 [40] is a schematic diagram of an ANN. An ANN consists of layers; an input layer, one or more hidden layers and an output layer. These layers consist of interconnected neurons (or nodes), which 'can be seen as computational units that receive inputs and process them to obtain an output' [41]. The connections between the neurons are weighted and these weight coefficients determine how the information passes through the network.

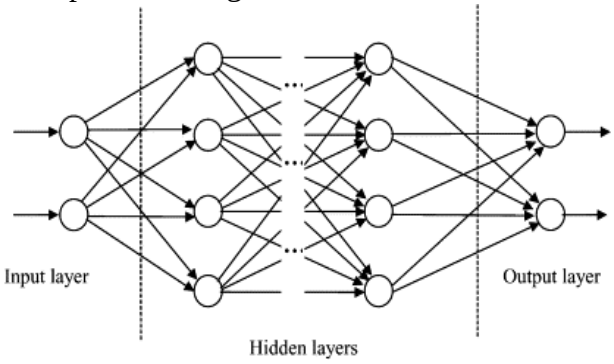


Figure 2.9 - Schematic diagram of a feed-forward artificial neural network [40]

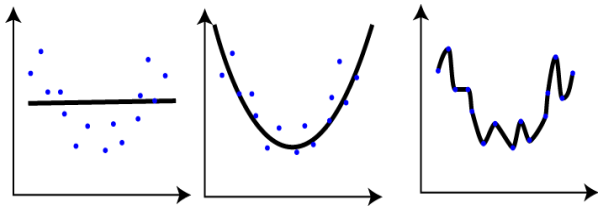


Figure 2.10 – A surrogate model with: a high bias (left image); a high variance (right image); and a suitable compromise between bias and variance (central image) [44]

2.3.2 Bias, Variance & Overfitting

The ANN is trained by providing a set of inputs and outputs to the network. The error between the computed output and the actual output is then minimised. Once the network has learned the data, it can then be used to predict the outputs for inputs that were not in the original training data. A network's ability to 'categorise correctly new examples that differ from those used for training is known as generalisation' [42]. For an ANN to generalise well, it must avoid being too simplistic or too flexible. If the network is too simplistic it is said to have a high bias. Bias 'quantifies the extent to which the surrogate model outputs differ from the true values' [43] and is given as an average over the whole data set. This scenario is shown in the left image in Figure 2.10 [44]. The opposite of bias is variance, which 'measures the extent to which the surrogate model is sensitive to a particular data set' [43]. A model with a high variance is too flexible, causing the network to learn each point individually; instead of learning the general trend of the data. This is known as overfitting and is shown in the right image of Figure 2.10.

2.3.2.1 Weight Decay Regularisation

Weight decay regularisation is one method that can be used to prevent overfitting. The weight coefficients determine how information is propagated forward through the network. There is a correlation between the magnitude of the weight coefficients and the flexibility of the network; i.e. large coefficients = overfitting. Thus overfitting can be prevented by limiting the growth of the weight coefficients [45]. During network training, the error function is minimised (the error function is the difference between the computed output and the actual output). Weight decay regularisation involves 'adding a penalty to the error function in order to discourage the coefficients from reaching large values' [42]. The

penalty function is the sum of the square of the weight coefficient magnitudes. The total error function to be minimised then becomes: $E_{TRAIN} + \lambda\Sigma(w^2)$. Where E_{TRAIN} is the training error, (the average error over the training data), $\Sigma(w^2)$ is the sum of the weight coefficient magnitudes squared and λ is the regularisation coefficient (also known as the weight decay coefficient). When λ is small, the weight coefficient magnitudes can be large, and the model will overfit the data. On the other hand, if λ is very large the weights coefficient magnitudes will be very small and the model will underfit the data (i.e. have a high bias). Figure 2.11 (below) is a spectrum of neural networks trained from $\lambda = 0$ (no regularisation) to $\lambda = \infty$. This figure shows how changing the value of λ can control overfitting in an ANN.

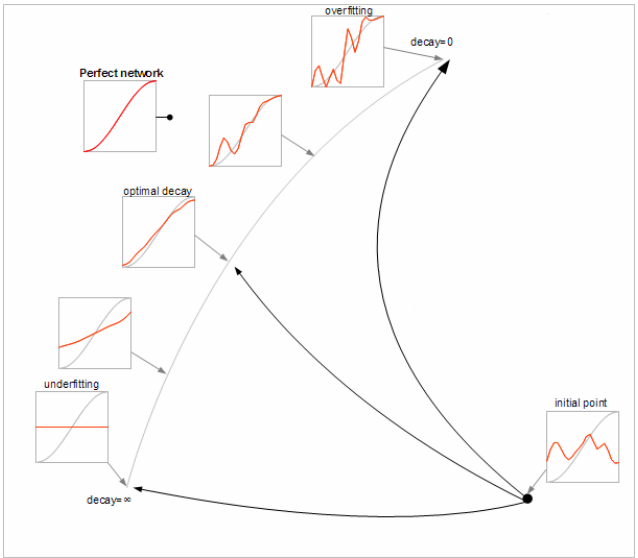


Figure 2.11 – Spectrum of neural networks trained with different values of λ – from zero value (no regularisation) to infinitely large λ [45]

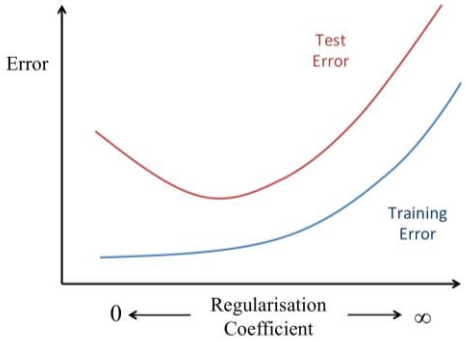


Figure 2.12 – Schematic of test and training error vs. regularisation coefficient (λ) for an ANN

2.3.2.2 Test and Training Error

In order to measure the effects of bias and variance, and to determine the correct value of λ to use, the test and training error for the model needs to be assessed. (The training error is the average error over the training data and the test error is the average prediction error over the independent test data). It is important to assess both the training error and the test error, as the training error will automatically decrease as the value of λ decreases. Whereas due to overfitting, there will be an optimum number of hidden units for the test error (see Figure 2.12). The training error is relatively straightforward to compute, as it is calculated by taking the average error over the training data. However, the test error is more difficult to determine and a variety of techniques can be used to calculate it. In this project the test error was calculated using Leave-One-Out Cross-Validation (LOOCV). LOOCV involves removing one point from the data set to use as the test data, and using the remaining points as the training data. The ANN is trained on the *n-1* data points, and the point that has been left out is used to test the accuracy of the network. This process is then repeated so that each point in the data is used once as the test data. More information on surrogate modelling can be found in *Pattern Recognition and Machine Learning* by Christopher Bishop [42].

3 Methodology

3.1 Group Methodology

The group was divided into three subgroups: automotive design, experimental design and HPC. The automotive design sub-group investigated the design and manufacture of a concept car using a combination of CAD, CFD and Additive Layer Manufacturing (ALM). The experimental sub-group redesigned and rebuilt an existing wind tunnel test chamber. The HPC sub-group investigated different aspects of high performance computing. This individual project was part of the High Performance Computing (HPC) sub-group. The overall aim of this group was to explore the use of HPC to create a VWT. In addition to the author, there were two other members in the HPC sub-group - James Walton and Luke Hamilton. James Walton was responsible for creating velocity profiles for an urban environment; with the aim of comparing these velocity profiles against classic wind tunnel flow profiles (with the design sub-group). Although there was no direct link with his project, the author worked with James, providing guidance on OpenFOAM and fluid mechanics; as he had no previous experience of CFD. The author also worked with Luke Hamilton to develop the VWT for the group project. The Ahmed body was chosen to develop the VWT due to the amount of experimental and computational literature available on the Ahmed body. The author's contribution to this task was to create a CFD model of flow around the Ahmed body - referred to in this project as the 'Ahmed body CFD model'. This CFD model needed to be relatively computationally inexpensive, but still retain a reasonable degree of accuracy, in order to run multiple simulations within the timeframe of this project. Consequently, this individual project focused on simpler CFD techniques such as `snappyHexMesh` and RANS turbulence modelling - to reduce computational cost of the model. Luke Hamilton's contribution was to investigate a more complex Ahmed body CFD model using Pointwise [46] and DES/LES turbulence modelling. The more complex Ahmed body CFD model was then used as the template for the VWT that was created for the concept car.

3.2 Individual Project Methodology

There were three primary work packages for this individual project. Firstly to create a computationally inexpensive Ahmed body CFD model. Secondly, to automate: the creation of model geometry; the meshing process; and the solution output in order to run multiple simulations to provide data to train a surrogate model. Thirdly, to create a surrogate model for the Ahmed body CFD model (referred to in this project as the 'Ahmed body surrogate model').

3.2.1 CFD Model Creation

The first phase of this project was the creation of the Ahmed body CFD model. This CFD model was then used to generate the data needed to train the ANN. In order to create a successful surrogate model of the Ahmed body it was essential that the Ahmed body CFD model had the right balance between accuracy and computational cost. If the model was erroneous, there may have been little or no correlation between the input and output parameters. Consequently there would be no pattern for the surrogate model to learn. On the other hand, if the model was very accurate but took weeks to run, it would not have

been able to create the amount of data needed to train the surrogate model. For surrogate modelling, the quality and quantity of data is important. Therefore the right balance had to be found when creating the Ahmed body CFD model. The CFD model was created using `snappyHexMesh` and OpenFOAM – the methodology of these processes is described in the following sections.

3.2.1.1 OpenFOAM

OpenFOAM was used to create all of the CFD models of the Ahmed body used in this project. Numerous Ahmed body CFD models were tried and validated against experimental results in order to determine appropriate parameters for the final CFD model, these included: mesh density, turbulence model and boundary conditions. The basic directory structure for an Ahmed body case in OpenFOAM can be seen in Figure 3.1 (below). When running an OpenFOAM case, the standard directory structure must be adopted. The CFD case is named (e.g. *`AhmedBody`*) and this directory contains all the information required to run the case. Within this directory there are three main sub-directories: *`constant`*, *`system`* and *`0`*. In these directories there are various data files that contain user-specified data known as 'dictionaries'. The *`constant`* directory contains the constant settings for the case. It includes details of the case mesh in the *`polyMesh`* sub-directory, and also contains information about the physical properties of the fluid. In this project the utility `snappyHexMesh` was used, and so a sub-directory called *`triSurface`* was included in *`constant`* that contained the geometry file of the Ahmed body (`snappyHexMesh` is discussed further in the next section). The *`system`* directory contains the simulation settings for the case. Inside this directory, the *`controlDict`* dictionary file contains run control parameters for: time control, data writing and run-time post-processing. Also in `system` there are two files called *`decomposeParDict`* and *`snappyHexMeshDict`*, these are dictionaries to control decomposition of the case (for running in parallel) and the meshing of the case (using `snappyHexMesh`). In the *`fvSchemes`* dictionary file, the discretisation schemes for the different terms of the Navier Stokes equations are specified and in *`fvSolution`* the solver settings and algorithm controls are set. Finally the *`0`* directory contains the initial conditions for the simulation at time = 0s. This includes the velocity, pressure and turbulence fields. As OpenFOAM runs a simulation, it saves the results to subsequent time directories - these results were post-processed using ParaView [47].

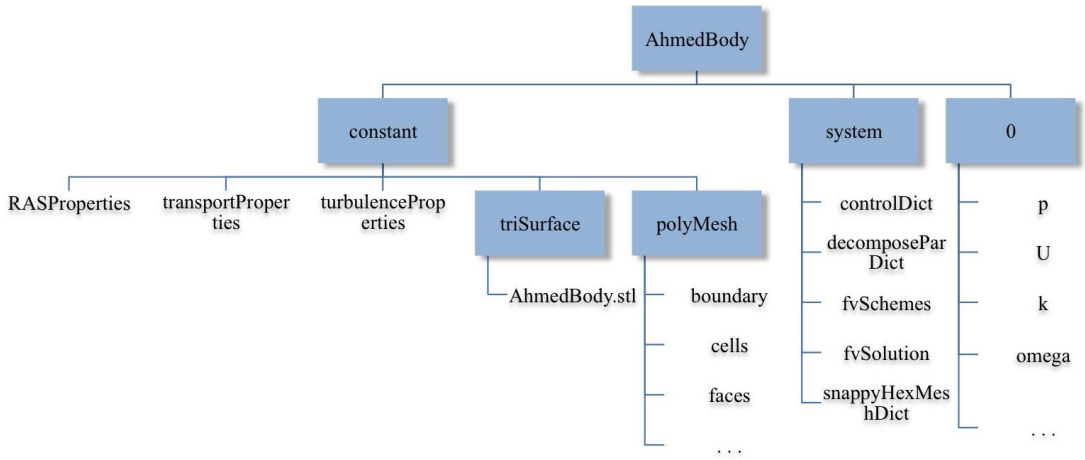


Figure 3.1 – OpenFOAM structure for Ahmed body case

3.2.1.2 snappyHexMesh

OpenFOAM contains an automatic mesh generator called `snappyHexMesh`, which 'generates 3-dimensional meshes containing hexahedra (hex) and split-hexahedra (split-hex) automatically from triangulated surface geometries in Stereolithography (STL) format' [48]. There are several steps involved in meshing with `snappyHexMesh`, however before this process can commence, a few precursor steps are required. The first step is to create the required geometry and save it in the *`triSurface`* directory of the OpenFOAM structure (see Figure 3.1). For this project, the geometry of the Ahmed body was created in SolidWorks [49] and exported as an STL file (see Figure 2.1 for the Ahmed body dimensions). The STL file had to be exported in ASCII format (rather than binary) for `snappyHexMesh` to work. When exporting, it was also found that it was important to check the box labeled *'do not translate STL output data to positive space'* to retain the position of the origin, otherwise the origin of the STL file moved once exported. Figure 3.2 is an image of the Ahmed body STL file created in SolidWorks. The second precursor step is to create a background hex mesh using `blockMesh`. There are several requirements when creating the background `blockMesh` as specified in the OpenFOAM user guide [48], namely: it must consist of more than one cell, be comprised purely of hex elements and the cell aspect ratio must be approximately 1. Figure 3.3 shows the `blockMesh` that was created in this step. In this project half the mesh was constructed and then a symmetry plane was applied to replicate a full mesh (to reduce computational expense). This is discussed in more detail later on in this report

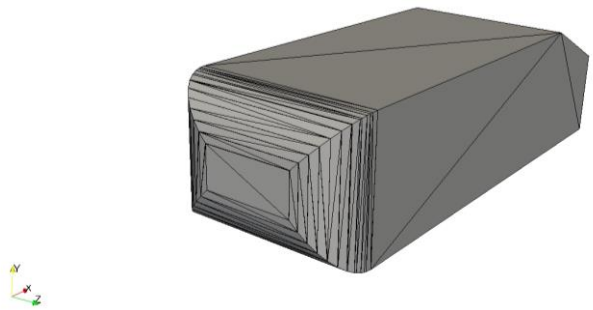


Figure 3.2 - Ahmed body STL file

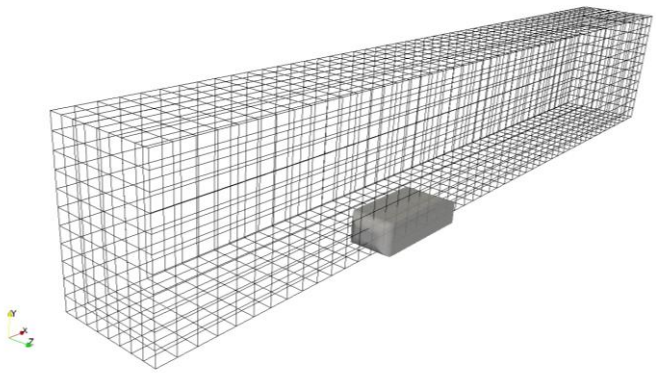


Figure 3.3 – Block mesh showing STL file

The following stages are automatically executed by `snappyHexMesh`, but will be outlined here for illustrative purposes. The first stage in the mesh generation process is the creation of a castellated mesh, which is executed in two parts. Initially, the `blockMesh` is refined around the surface of the STL file and afterwards the unwanted cells are removed. Figure 3.4 illustrates how the mesh is refined. In addition to refining the mesh around a surface, the mesh can also be refined inside a closed region such as a box.

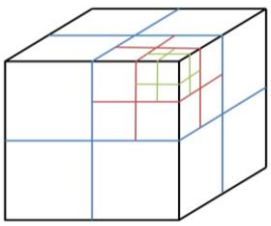


Figure 3.4 – Mesh refinement for one cell: level 1, level 2 and level 3

When using snappyHexMesh, the user specifies whether the mesh is an internal one (i.e. for flow in a pipe) or an external one (i.e. for flow around an object) in the *snappyHexMeshDict* file, by specifying a point inside the mesh. In this case the mesh is an external one, so the cells inside the STL file are removed. Figure 3.5 shows the castellated mesh around the Ahmed body. In the second stage of the process the vertex points of the castellated mesh are snapped onto the surface of the STL file (hence the name snappyHexMesh) - this stage can be seen in Figure 3.6. The third (optional) stage of snappyHexMesh is to add layers of hexahedral cells to wall surfaces. In this step, the mesh on the surface is inflated in the direction normal to the surface to create layers of hexahedral cells. This process ensures a good quality mesh on these surfaces to capture the complex boundary layer flow that occurs at a wall.

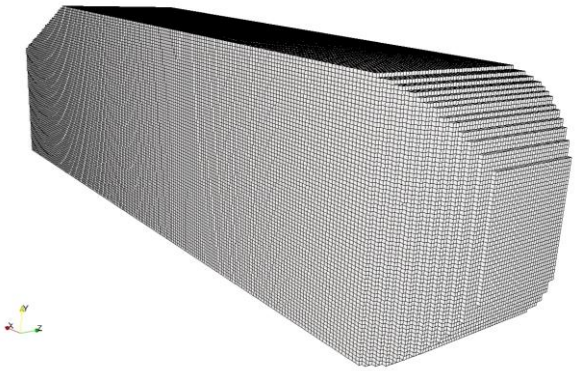


Figure 3.5 – Castellated mesh around the Ahmed body

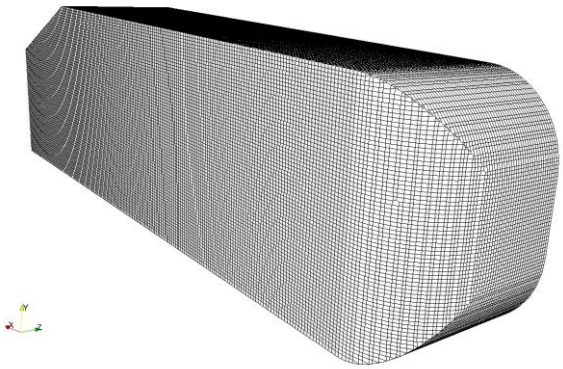


Figure 3.6 – Snapped mesh around the Ahmed body

One significant advantage of using snappyHexMesh is that it is a robust meshing tool, which can consistently create good quality meshes. In *snappyHexMeshDict* there is a sub-dictionary called meshQualityControls, which contains pre-specified mesh quality settings. During the generation of a mesh, the mesh quality is constantly monitored, and if during a step in the meshing process the mesh violates these quality controls, then that step is reversed before proceeding. (If the mesh cannot be created then the quality controls are relaxed). The result is a high-quality, hex dominated mesh. In this project, the default mesh quality controls were used as specified in the OpenFOAM user guide [50]. After a mesh had been created, the utility checkMesh was used to check the quality of the mesh and to print various mesh statistics to screen, including: the number of cells, the maximum aspect ratio, the mesh non-orthogonality, the maximum skewness and whether the mesh has failed any mesh checks. Mesh orthogonality is defined as; the angle between the vector connecting two adjacent cell centres and the normal vector of the face connecting the two cells - a perfect cell has a value of 0. The aspect ratio is the ratio of a cell's longest edge to its shortest edge and has a value of 1 for a perfect cell. Skewness is a measure of the distance between the point where the vector between two cell centres intersects the face connecting the cell centres, and the centre of the connecting face. The checkMesh utility was used for all of the meshes generated in this project and the orthogonality, aspect ratio and skewness of the mesh was checked to ensure the meshes were of suitable quality.

3.2.2 CFD Model Automation

The second phase of this project was to develop a technique to automate the running of multiple Ahmed body CFD models. In this project, OpenFOAM was run using Linux - an

open-source operating system based on UNIX. Linux is controlled using a shell, which is defined as 'any program that takes input from the user, translates into instructions that the operating system can understand, and conveys the operating system's output back to the user' [51]. Linux is primarily operated using a command-line user interface, which makes it easier to automate via scripting than a graphical user interface. A script is a file that contains shell commands that can be executed by running the script file instead of manually typing the commands. Bash scripting was used to automate the CFD model creation and simulation in OpenFOAM. Bash is a command-line user interface shell commonly used with Linux. To automate the CFD simulations two script files were written. Firstly a bash script was created that meshed, ran and post-processed one CFD simulation. Once this script had been successfully implemented, a second bash script was written to automate the running of multiple CFD simulations. These scripts were developed using very quick (and completely inaccurate) CFD models of the Ahmed body that took less than a minute to run, enabling fast progress to be made. After it had been established that the scripts were running as desired, they were then used to run the final Ahmed body CFD models.

3.2.3 Project Management

The primary project management tool used in this project was a Gantt Chart. This was created at the beginning of the project and is discussed in *Individual Report - I1* [2]. The project was initially split into three main areas: preliminary work; initial surrogate model creation; and Ahmed body surrogate model creation. The preliminary work mainly involved learning OpenFOAM by completing various tutorials, and was undertaken in the first term. In the second term, the original plan was to create an initial surrogate model using a quick and simple CFD case (the flow around a cube) before moving on the Ahmed body. Despite planning, this project evolved throughout its duration and thus the project plan developed as the project progressed. The following section outlines the significant changes that were made to the original project plan.

3.2.3.1 Project Development

The original intention was to initially create a simple CFD model of the flow around a cylinder to replicate an experiment that was undertaken by the experimental sub-group in November. However, the results from their cylinder experiment were in poor agreement with theory, which the experimental group concluded was due to inaccuracies within their experimental measuring equipment [52] [53]. Given that the wind tunnel was not suitable for validation (before the test section had been rebuilt), all validation was done using experimental literature of the Ahmed body.

It was also found that there were several issues with creating an initial surrogate model based on a cube. Firstly, despite being a simple shape, the flow around a cube is very complex (which is difficult to model). Secondly, for the cube there are only three geometric parameters that could be varied (height, width and depth) and varying these three dimensions may not have resulted in significant changes in the drag coefficient. Thus there may not have been a pattern for the surrogate model to predict. Due to these reasons, this idea was omitted and the surrogate model was developed using the Ahmed body.

The most significant change to the project plan was the time spent on developing the Ahmed body CFD model. As discussed, it was essential to achieve the right balance between computational expense and accuracy. As the project progressed, it became

apparent that the flow around the Ahmed body was significantly more complex than expected. Consequently, far more effort was spent on producing the basic CFD model than originally intended. Due to these unexpected project delays, the expertise of Richard Everson (Professor of Machine Learning at the University of Exeter) was called upon to create a surrogate model for the Ahmed body in order to assess the feasibility of using a surrogate model in automotive design.

3.2.3.2 Health & Safety

This individual project was entirely office based, therefore the primary health and safety hazards were posed by spending prolonged periods using Display Screen Equipment (DSE). To mitigate the risk posed by DSE, care was taken to ensure a correctly positioned workstation (following DSE guidelines provided by the University of Exeter [54]) and regular breaks were taken. A DSE user self-assessment form was downloaded from the University of Exeter's website and completed at the beginning of this project [55] – see Appendix A. This form showed that the workstation used in this project was acceptable and no further action was required.

4 Ahmed Body CFD Model

A CFD model of the Ahmed body was created using OpenFOAM. This section of the report details how the final design of the CFD model was reached; presents the results from the final CFD model; and analyses these results.

4.1 Design

4.1.1 Geometry

The dimensions for the wind tunnel geometry were taken from the ERCOFTAC workshop on the Ahmed body [14] and can be seen in Figure 4.1 below. ERCOFTAC guidelines state that if the simulation is steady-state case, then a half body can be modeled with a symmetry plane to imitate a full body scenario - this approach was adopted for this project. The computational domain had a width of 0.935m and a height of 1.4m. To ensure that neither the inlet nor the outlet conditions affected the flow around the Ahmed body, the domain was two body lengths in front of the Ahmed body, and extended to five body lengths behind the Ahmed body. A symmetry boundary condition was employed along the symmetry plane (see Figure 4.1).

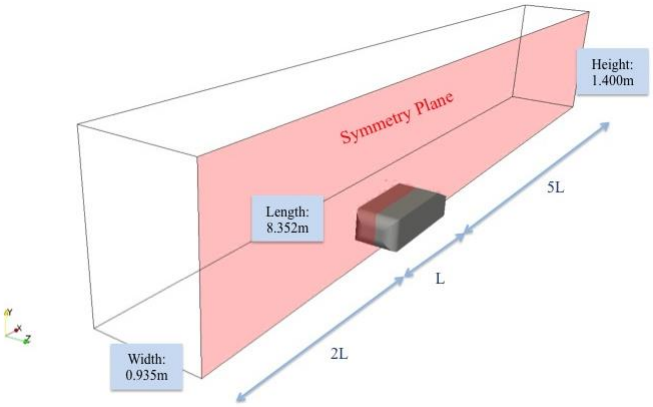


Figure 4.1 – Virtual wind tunnel geometry

4.1.2 SnappyHexMesh

4.1.2.1 Resolving Sharp Edges

As described in the methodology section, `snappyHexMesh` was used to generate the mesh for the CFD Ahmed body model. The first issue that was encountered when meshing the CFD Ahmed body model was that `snappyHexMesh` struggled to correctly resolve sharp corners. When modelling the Ahmed body it is essential to correctly resolve sharp edges as it is this geometry that is responsible for separation on the rear slant surface and the creation of trailing vortices. To overcome this issue, the `surfaceFeatureExtract` utility was used. This utility extracts feature edges from the geometry file (the desired features are specified in *`snappyHexMeshDict`*) and then points on the mesh are aligned with these feature edges. Figure 4.2 and Figure 4.3 are images of the mesh around the rear face of the Ahmed body and the slanted surface, without and with the `surfaceFeatureExtract` utility, respectively.

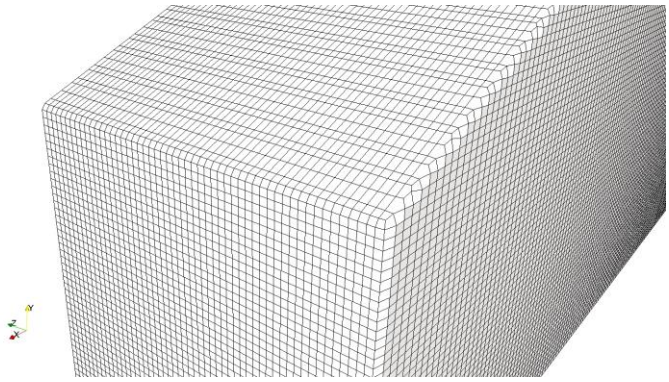


Figure 4.2 – Mesh around the rear of the Ahmed body without `surfaceFeatureExtract`

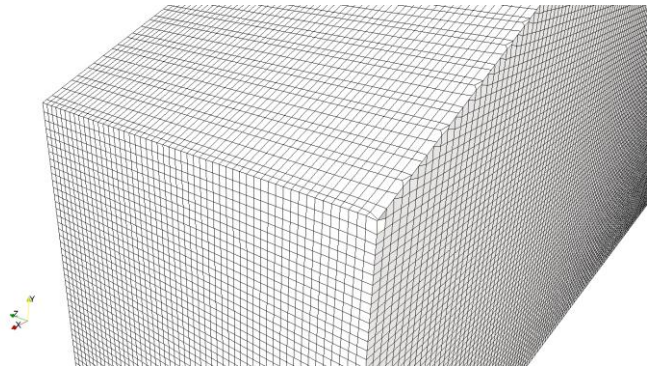


Figure 4.3 - Mesh around the rear of the Ahmed body with `surfaceFeatureExtract`

4.1.2.2 Boundary Layer Meshing

As discussed in the background section of this report, the mesh in the near-wall region is essential to achieving accurate results. It is possible to create a boundary layer mesh with the third (layer addition) stage of `snappyHexMesh`. This process was applied to the Ahmed body surface, however it was found that the boundary layers failed to generate in certain regions and did not extend all the way to the symmetry plane. A detailed investigation into the layer addition step of `snappyHexMesh` was undertaken, and multiple parameter combinations were tried in the layer addition section of *`snappyHexMeshDict`*. Varying the parameters that controlled the layer addition procedure, made an improvement to the boundary layer mesh that was generated. However, trying to determine the optimum parameter values was difficult, because there did not appear to be a logical correlation between the parameter values and the quality of the boundary layer – it seemed to be more of a random process. Figure 4.4 (below) is a section view of the mesh showing the best quality boundary layer that was created. However, even this boundary layer has not been generated around the corners or up to the symmetry plane. As the boundary layer mesh was not successful around the corners, it was thought that the `surfaceFeatureExtract` utility might be the cause of the problems. However, the boundary layer mesh failed to successfully generate around the corners, even without using `surfaceFeatureExtract`.

Next, the `refineWallLayer` utility was used to investigate whether it would be able to produce a suitable boundary layer mesh. This utility cuts the cells next to a boundary by a certain factor specified by the user (i.e. a factor of 0.5 will cut the cells in half). Although this utility gave a greater control over the height of the first cell in the near-wall mesh, it did not improve the quality of the mesh in this region. After all other avenues had been exhausted, Eugene de Villiers (who is listed as one of the creators of `snappyHexMesh`) was contacted[1] to see if he could provide any assistance. Unfortunately, he stated that `snappyHexMesh` has some serious issues and his advice was to use an alternative meshing software to create boundary layers. However, this was not feasible for the project as the aim was to create the whole model within OpenFOAM to allow for the automation of the CFD model creation.

[1] Personal communication [email] 4/02/2014

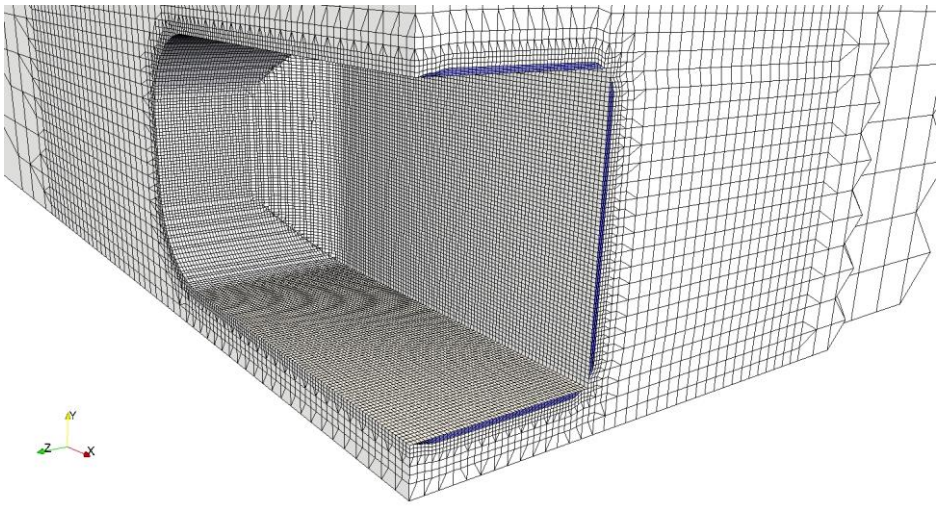


Figure 4.4 - Section of mesh showing the boundary layer mesh that was created using `snappyHexMesh` (highlighted in blue)

4.1.3 Boundary Conditions

In the investigations by Lienhart et al. [13] and Meile et al. [11] the experiments were carried out in a ¾ open test section wind tunnel. The Ahmed body CFD model was created to replicate these conditions, with the top and the outer side of the model set as symmetry conditions. A symmetry boundary condition was also applied at the symmetry plane. At a symmetry boundary, the following conditions apply: the wall shear stresses are zero; the velocity component normal to the boundary is zero; and the flux through the boundary is zero. The floor of the model and the surface of the Ahmed body were set as wall boundaries with a 'no-slip' condition. The inlet was set as a velocity inlet with a uniform flow profile and a velocity of 40 m/s in the x-direction. The outlet was set as a pressure[2] outlet at 0 m^2/s^2 and the kinematic viscosity of air was set at 1.5×10^{-5} m^2/s. Two RANS turbulence models were investigated – the k-ω SST model and the Spalart Allmaras (S-A) model. The following sections outline these two turbulence models in more detail.

4.1.3.1 k-ω SST Turbulence Model

The k-ω SST model is a two-equation model, which solves a transport equation for k (turbulent kinetic energy) and ω (specific dissipation). The k-ω SST turbulence model is effectively a hybrid of the standard k-ε and k-ω models. It uses the standard k-ω turbulence model in the near-wall regions, and the standard k-ε model in the free stream. This means it can be used in wall resolved approaches (unlike the standard k-ε model) whilst avoiding the problems caused by sensitivity to free-stream turbulence inlet properties that the standard k-ω model experiences. There are several methods used to approximate the turbulent boundary conditions, in this project the following equation was used to estimate the turbulent kinetic energy [56]:

Equation 3

$$k = \frac{3}{2}(UI^2)$$

[2] In OpenFOAM the pressure is scaled by the density – hence the units m^2/s^2

Where: k – turbulent energy (m^2/s^2); U – mean flow velocity (m/s); I – turbulent intensity

ERCOFTAC recommends that the specific dissipation is calculated based on the measured inlet viscosity ratio [14], therefore the following equation was used to estimate ω [56]:

Equation 4

$$\omega = \frac{k}{\nu}\left(\frac{\mu_t}{\mu}\right)^{-1}$$

Where: ω – specific dissipation (s^{-1}); ν – kinematic viscosity (m^2/s); μ_t/μ - viscosity ratio

In the experimental study by Lienhart et al., the measured turbulent intensity of the wind tunnel used in their experiments was less than 0.25% [13]. Thus for Equation 3, the turbulent intensity was set at 0.2%, which gave a value for k of 9.6×10^{-3} m^2/s^2. The measured inlet viscosity ratio was approximately 10 [14], and this was used in Equation 4 to give a value for ω of 64 s^{-1}.

4.1.3.2 Spalart Allmaras Turbulence Model

The Spalart Allmaras turbulence model is a one-equation model which solves a transport equation for the turbulence variable $\tilde{\nu}$ (nutilda). The boundary conditions for this model are [57]:

Equation 5

$$\tilde{\nu}_{wall} = 0$$

Equation 6

$$\tilde{\nu}_{freestream} = 3\nu_\infty \text{ to } 5\nu_\infty$$

Where: ν_∞ - kinematic viscosity of the free stream fluid (m^2/s)

Using Equation 6, the value of $\tilde{\nu}$ was estimated by setting this parameter to five times the free stream kinematic viscosity (7.5×10^{-5} m^2/s).

4.1.3.3 Wall Functions

For incompressible RAS calculations, the choice of wall function model specified through ν_t in the *0/nut* file [58]. The user can specify this file directly (as is the case with the S-A turbulence model) or OpenFOAM will generate it automatically using information specified in the other parameters in the *0* directory. The `nutkWallFunction` and the `nutUSpaldingWallFunction` were used for the k-ω SST model and the S-A model, respectively.

4.1.4 Solution Set-up

In the *fvSchemes* dictionary, the discretisation schemes for the different terms of the Navier Stokes equations are specified. In the simulations for this project, the time derivative term was set as `steadyState` (i.e. the time derivative terms are not solved). The discretisation scheme for the divergence, gradient and laplacian terms was specified as `Gauss` (it can only be `Gauss` for divergence and laplacian schemes). The `Gauss` discretisation scheme is 'standard finite volume discretisation of Gaussian integration which requires the interpolation of values from cell centres to face centres' [59]. Thus when using the `Gauss` discretisation method, the interpolation scheme must also be

specified. For the gradient and laplacian terms, the interpolation method was specified as `linear` (this is a central differencing, 2nd order scheme), and the interpolation scheme for divergence was specified as `upwind` (this is a 1st order upwind method). The `Gauss` discretisation scheme also requires a surface normal gradient scheme for the laplacian terms – this was specified as `corrected`. The surface normal gradient scheme is a modification to the differencing scheme that accounts for non-orthogonal meshes (i.e. meshes where the gradient between two cell centres at the face which connects the cell centres does not equal 0).

The discretisation of the Navier Stokes equations results in a system of algebraic equations. In OpenFOAM there are different solvers available that can be employed to solve these equations - these are specified in *`fvSolution`*. In this project, the `PCG` (preconditioned conjugate gradient) solver for symmetric matrices was selected for the pressure variable and the `PBiCG` (preconditioned bi-conjugate gradient) solver for asymmetric matrices was selected for the velocity and turbulence variables. These solvers use the conjugate gradient method to iteratively solve linear sets of equations, with a preconditioner that accelerates the process. More information on solvers and discretisation methods can be found in *Computational Fluid Dynamics: A Practical Approach* (Chapter 4: CFD Techniques) [60]. Also in *`fvSolution`*, OpenFOAM allows the user to specify a tolerance of the residuals for each variable being solved – if the simulation residuals reach this value then it stops running. In this project, residual tolerance was set at 1×10^{-6} for the residuals of pressure, velocity, nuTilda, k and ω.

Before the CFD simulations were run, the fields in the *`0`* directory were initialised using `potentialFoam`. This solver can improve solution convergence by solving the potential flow equations to generate initial fields for the simulation. The simulations were then run using `simpleFoam` – the steady-state solver for incompressible turbulent flow.

4.1.5 Initial Investigations

4.1.5.1 Strut Influence

In the experimental studies of the Ahmed body in literature, the Ahmed body is mounted on struts. In general, when the drag force is calculated, the drag on the struts is omitted from this calculation. This is the case for the original experiments by Ahmed et al. [4] and the experimental study by Meile et al. [11] (Lienhart et al. [13] did not calculate the drag coefficient). Hence in their guidelines, ERCOFTAC states that the struts can be removed from the CFD simulations [14]. Omitting the struts from the CFD simulations was preferable, as it simplified the mesh. Nonetheless, before the struts were omitted from the CFD simulations, an investigation was performed into the influence of the struts on the drag coefficient. Firstly, the Ahmed body was meshed (including the struts) and a mesh convergence study was performed. The number of elements in the mesh was increased, and the drag coefficient was recorded to determine a suitable mesh density for the strut investigation. The drag coefficient was found using function objects in OpenFOAM. Function objects allow the user to 'carry out post processing automatically whilst the simulation is running' [61]. To calculate the drag force, the function object `forceCoeffs` was used. This calculates the drag coefficient and writes it to a file. A refinement box around the Ahmed body was used to reduce the computational expense of the mesh. This technique involves having a finer mesh in the more complex flow regions and having a coarser mesh for areas of flow in less important regions. The mesh density was increased by increasing the refinement level in the box around the Ahmed body.

The mesh convergence study indicated that a suitable mesh density was around 2.5 million elements. Three cases were then run using this mesh density, using the boundary conditions and solution set-up as specified in sections 4.1.3 and 4.1.4 above, with the k-ω SST turbulence model. In the first case, the Ahmed body had struts and the struts were included in the drag coefficient calculation. The second case had struts, but the struts were not included in the drag coefficient calculation. This was achieved by creating two STL files (one for the body and one for the struts) to omit the struts from the drag coefficient calculation. For the third case, the Ahmed body had no struts (but was raised off the ground). Table 1 below shows the results from this study. These results show that if the struts are included in the drag calculation then the drag coefficient increases by around 12%. These results demonstrate that despite being a small part of the Ahmed body, the drag experienced by the struts is relatively high - this is due to their cylindrical shape. Table 1 also shows that the difference between, including the struts in the simulation whilst omitting them from the drag calculation, and simply omitting the struts altogether, is negligible. Therefore it was decided to omit the struts from subsequent Ahmed body CFD models.

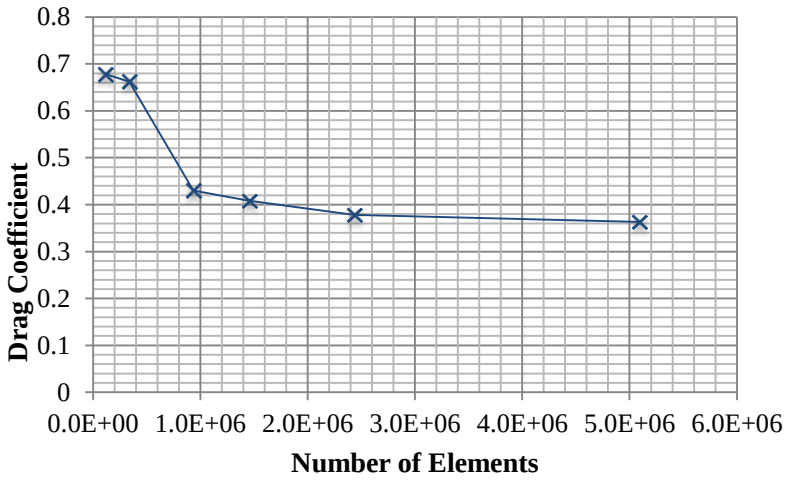


Figure 4.5 – Mesh convergence study on Ahmed body with struts

Table 1 – Influence of struts on drag coefficient

CFD Model	Drag Coefficient
Struts included in drag coefficient	0.386
Struts not included in drag coefficient	0.343
No struts	0.344

4.1.5.2 Turbulence Model Study

In previous computational studies of the Ahmed body, the k-ω SST RANS turbulence model has performed well in comparison with other standard RANS turbulence models (k-ε, RNG k-ε and realisable k-ε) [24]. Therefore it was decided to use this model for the CFD Ahmed body model along with the S-A turbulence model - as this model was developed for aerodynamic flow [62]. Other turbulence models were not investigated as Luke Hamilton was investigating more accurate turbulence modelling methods. A mesh convergence study was performed on the Ahmed body without struts and with a rear slant angle of 25° (see Figure 4.6 for a graph of the results). This mesh convergence study was performed in a similar manner to the previous mesh convergence study – by including a refinement box and increasing the level of refinement inside this box. Figure 4.7 is a graph

of the computational expense (measured in CPU hours) plotted against the number of elements in the mesh. The aim of a mesh convergence study is to determine the coarsest mesh that gives a mesh independent solution; in order to reduce computational expense. A solution tolerance of 3% was chosen to determine when a solution had become mesh independent.
Figure 4.6 shows that for both turbulence models, the solution converges at a mesh density of around 2.5 million elements. For both models increasing the mesh density from 2.5 million to 4.4 million elements changed the value of the drag coefficient by less than 3%, but resulted in a computational increase of over 100%.

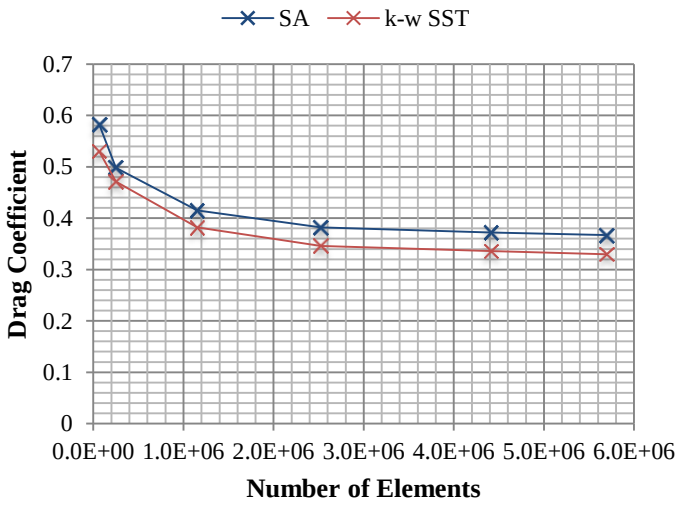


Figure 4.6 –Mesh convergence study for k-ω SST and S-A turbulence models

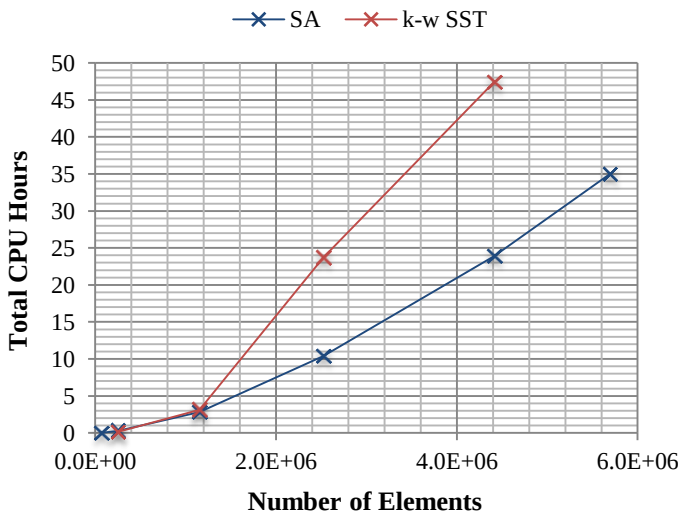


Figure 4.7 – Computational expense of k-ω SST and S-A turbulence models at different mesh densities

Figure 4.7 shows that the k-ω SST turbulence model is more computationally expensive than the S-A turbulence model. The simulations using the S-A model took less time to run because they reached the residual tolerance criteria (1×10^{6}) within 1,000 iterations. For the k-ω SST model at mesh densities greater than 1 million elements, the residuals did not reach the tolerance criteria. Instead they plateaued at a much higher values (the pressure residuals were around 4×10^{-3}) and subsequently these simulations were run for 2,000 iterations.

Using a mesh density of 2.5 million elements, Ahmed body CFD simulations were run at rear slant angles of 25° and 35° for both turbulence models. The results were compared with the results from Meile et al. [11] and can be seen in Table 2. This table shows that the k-ω SST model is more accurate than the S-A turbulence model, with a percentage error of 15.1% compared to 27.8% for a rear slant angle of 25°. However, neither turbulence model showed a drop in drag coefficient at 35°. This drop is caused by the flow separating from the rear slant surface; velocity profiles for both models indicated that the flow was still attached at 35°.

Table 2 - Drag coefficient results for k-ω SST and S-A turbulence models

Angle (°)	Drag Coefficient				
	Meile et al. [11]	k-ω SST	S-A	% Error k-ω SST	% Error S-A
25	0.299	0.344	0.382	15.1	27.8
35	0.279	0.402	0.428	44.1	53.4

Simulations were then run at rear slant angles of 10°, 25°, 35° and 40° and the flow in the wake region was assessed – see Appendix B for the results. It was observed that the S-A predicted flow separation at 40° (and a subsequent decrease in drag coefficient), while even at 40° the flow was still fully attached to the rear slant surface in the k-ω SST model. It was also observed that the S-A model was better at predicting the general flow behaviour in the wake region of the Ahmed body. It was suspected that a low quality mesh on the Ahmed body (particularly on the rear slant surface) was causing the poor performance of both the turbulence models. As previously discussed, a boundary layer mesh is normally employed on the surface of the Ahmed body to ensure a good quality mesh on this surface to accurately model the flow in the near-wall region. However, `snappyHexMesh` failed to produce a satisfactory boundary layer mesh and so this meshing technique could not be employed here.

4.1.5.3 Outflow Study

It was suggested by Gavin Tabor (the project supervisor) that one possible cause of the error between the CFD results and experimental results was due to the symmetry conditions imposed at the top and outer side of the model. In the experimental studies [11,13], the wind tunnel had a ¾ open test section (i.e. it had a floor but no walls). Employing a symmetry boundary condition at the sides of the wind tunnel is not a true reflection of an open wind tunnel, as it effectively imposes a no-slip wall at the sides. In effect this will channel the flow around the Ahmed body, causing an increase in flow speed (and consequently an increase in drag) around the Ahmed body compared to the experimental conditions. Hence, this boundary condition was changed from symmetry to outflow for the top and outer side of the wind tunnel, and the CFD simulations were rerun for the Ahmed body at a rear slant angles of 10°, 25°, 35° and 40° for both turbulence models. While this had no discernible impact on the flow behaviour around the Ahmed body itself (the k-w SST model still failed to produce separated flow), it did cause an average decrease in drag coefficient for both models of around 7.5%.

As mentioned, it was suspected that the low quality mesh at the boundary layer of the Ahmed body was causing the poor performance of both turbulence models. Consequently, the work completed so far for this project was passed onto Luke Hamilton who was using the commercial meshing software; Pointwise. It was decided that he would investigate RANS turbulence modelling to see if using a better quality mesh (generated by Pointwise) would improve the accuracy of the results.

4.2 Results & Analysis

The S-A turbulence model was chosen for the final CFD model, as this turbulence model was better at predicting the flow behaviour around the Ahmed body. The S-A model was also able to predict fully separated flow at larger rear slant angles, which is one of the key features of the drag coefficient behaviour. Furthermore, this model was also less

computationally expensive than the k-ω SST model, thus making it the preferable choice for running multiple simulations. The geometry, solution set-up, and boundary conditions were the same as specified in the preceding sections (including outflow conditions at the top and side of the model). The images illustrating the final model in this section are all taken from an Ahmed body with a rear slant angle of 25°. The final mesh had a refinement level of 6 around the surface of the Ahmed body. The mesh also included a refinement box that was ½L wide, ½L high and extended to 1.5L behind the Ahmed body with a refinement level of 4. Figure 4.8 and Figure 4.9 show the mesh used for the final CFD model. The `checkMesh` utility was run and the mesh statistics for this mesh were as follows:

- *Number of cells:* 2,741,553
- *Max aspect ratio:* 3.9754 OK
- *Mesh non-orthogonality:* Max: 42.2696 average: 5.57615
    - Non-orthogonality check OK
- *Max skewness:* 1.82828 OK
- Mesh OK

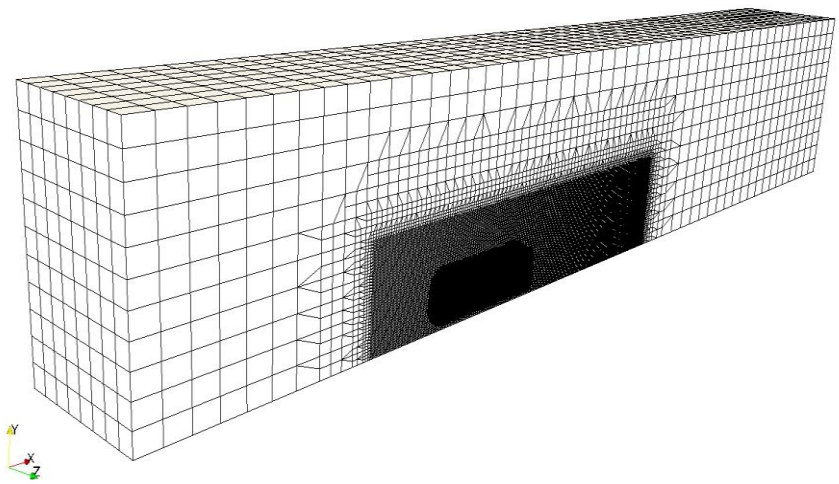


Figure 4.8 - Mesh for final Ahmed body CFD model

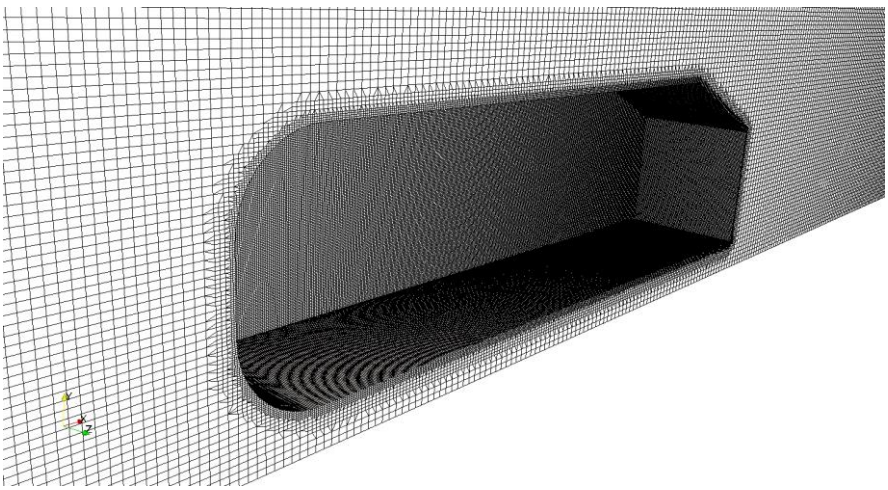


Figure 4.9 - Mesh around the Ahmed body for final CFD model

As wall functions were employed, the y^+ values for this mesh were checked to ensure they were within the acceptable range of 20 – 200 [31]. Figure 4.10 displays the range of y^+ values for the mesh on the Ahmed body surface (this figure has been reflected to visualise a full Ahmed body). These figures show that the y^+ value is at a minimum at the centre of the front of the Ahmed body (where the flow stagnates), and in the wake region where the flow has separated from the Ahmed body surface. The y^+ values are at a maximum around the curved front of the Ahmed body where the flow is accelerated over the surface. This is to be expected, as the y^+ values will be at a minimum where the flow velocity is at a minimum, and at a maximum where the flow velocity is at a maximum. The average y^+ value was 95 for the mesh on the Ahmed body. It would have been preferable to have y^+ values closer to 20 at the Ahmed body surface, but without a boundary layer mesh it was impossible to have sufficient control over the y^+ values to achieve this.

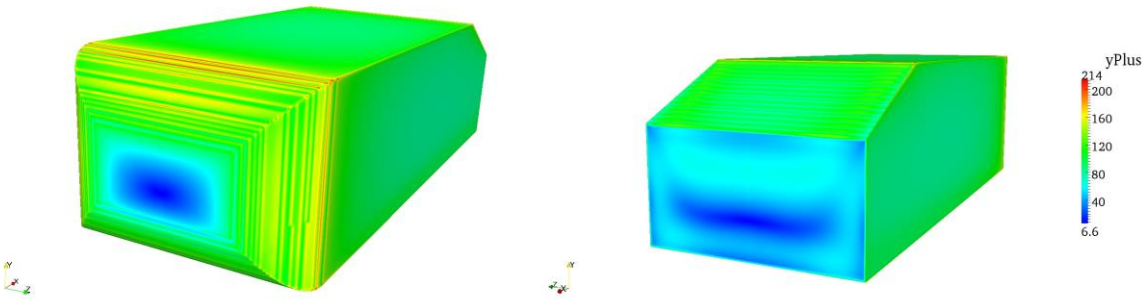


Figure 4.10 – y^+ values for Ahmed body surface

4.2.1 Model Verification

To determine the accuracy of the final CFD model, in order to assess its suitability for use in the creation of a surrogate model, verification and validation assessments were performed. A verification assessment of the model 'determines if the programming and computational implementation of the conceptual model is correct' [63]. The results provided by the final Ahmed body CFD model were checked to ensure that they had converged. Convergence was assessed by viewing the residual values for the simulation (see Figure 4.11) and by monitoring the drag coefficient (see Figure 4.12). Figure 4.11 shows that all the residuals for the CFD simulation have reached the convergence tolerance that was set at 1×10^{-6}, indicating that the final result has converged. Figure 4.12 is a plot of the drag coefficient against iteration number. This graph demonstrates that the drag coefficient converges at around 600 iterations. Both these figures indicate that the solution has converged and reached a steady-state solution. In addition to checking the residuals and monitoring values of interest, it is also important to determine that the solution is independent of the mesh using a convergence study; this was completed in section 4.1.5.2.

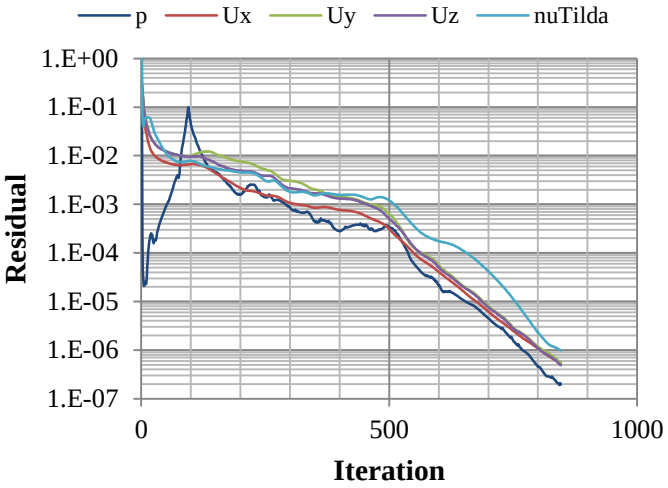


Figure 4.11 – Residual values for the final Ahmed body CFD model

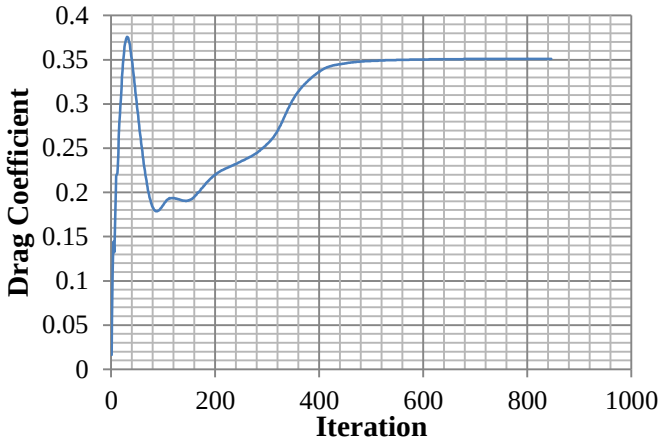


Figure 4.12 - Drag coefficient vs. iteration number for the final Ahmed body CFD model

4.2.2 Model Validation

A model validation assessment 'determines if the computational simulation agrees with physical reality' [63]. For this CFD model, three validation assessments were carried out; the drag coefficient values were compared with experimental data from Meile et al. [11]; the wake flow profiles were compared with experimental data from Lienhart et al. [13]; and the relationship between drag coefficient and rear slant angle was compared with experimental data from Ahmed et al [4].

4.2.2.1 Drag Coefficient

The drag coefficient was compared with experimental data from Meile et al. – the results are shown in Table 3 below. These results show that the S-A model over-predicts the drag coefficient. The error is particularly high at $\alpha = 35°$ as the flow should separate from the rear slant surface at this angle, causing a decrease in drag coefficient. However, for the S-A turbulence model, the flow did not separate until $\alpha = 40°$.

Table 3 – Drag coefficient values for final CFD model

Rear Slant Angle (°)	Drag Coefficient		
	S-A	Meile et al. [11]	% Error
25	0.351	0.299	17.4
35	0.395	0.279	41.6

4.2.2.2 Wake Flow

The wake flow at a rear slant angle of 25° was compared with experimental data from Lienhart et al. [13] – see Figure 4.13 and Figure 4.14 (results have been reflected for the CFD model to illustrate a full case). The development of the counter-rotating trailing vortex system was assessed by plotting the velocity vectors and magnitude at four planes in the x-normal direction. These planes were at x = 0mm, 80mm, 200mm, 500mm, where x = 0 is at the rear of the Ahmed body. The results shown in Figure 4.14 compare well with the experimental results in Figure 4.13, demonstrating that the CFD model has correctly predicted the presence of the counter-rotating trailing vortex system in the wake flow of the Ahmed body at $\alpha = 25°$.

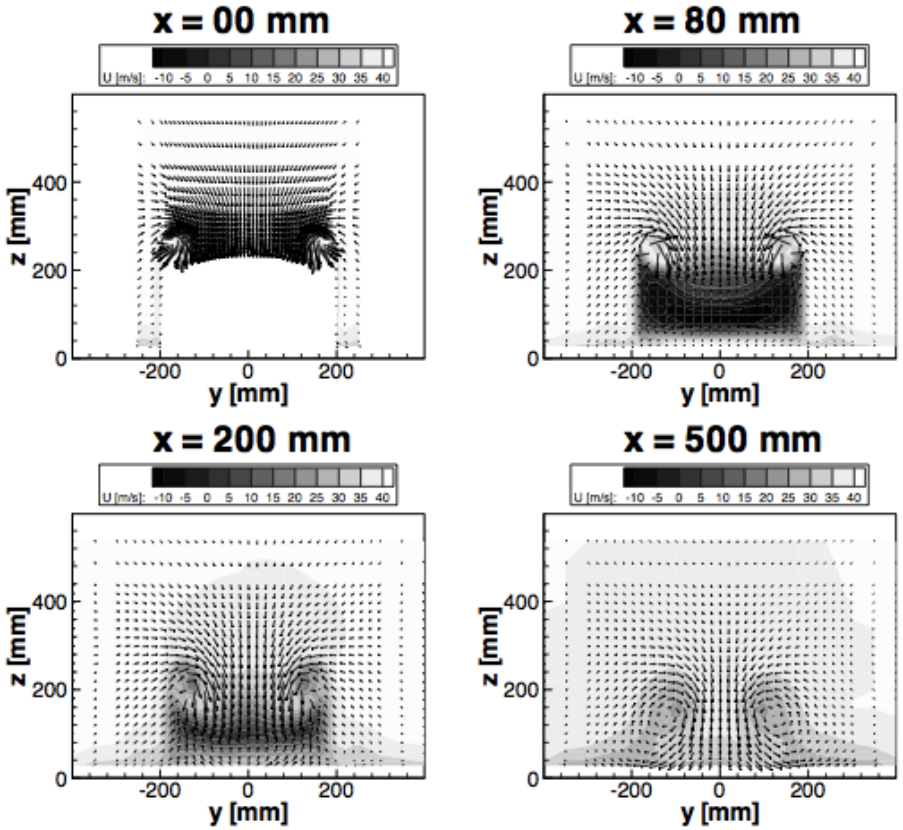


Figure 4.13 - Development of the counter-rotating trailing vortex system within the wake of the 25° slant Ahmed body, Lienhart et al. [13]

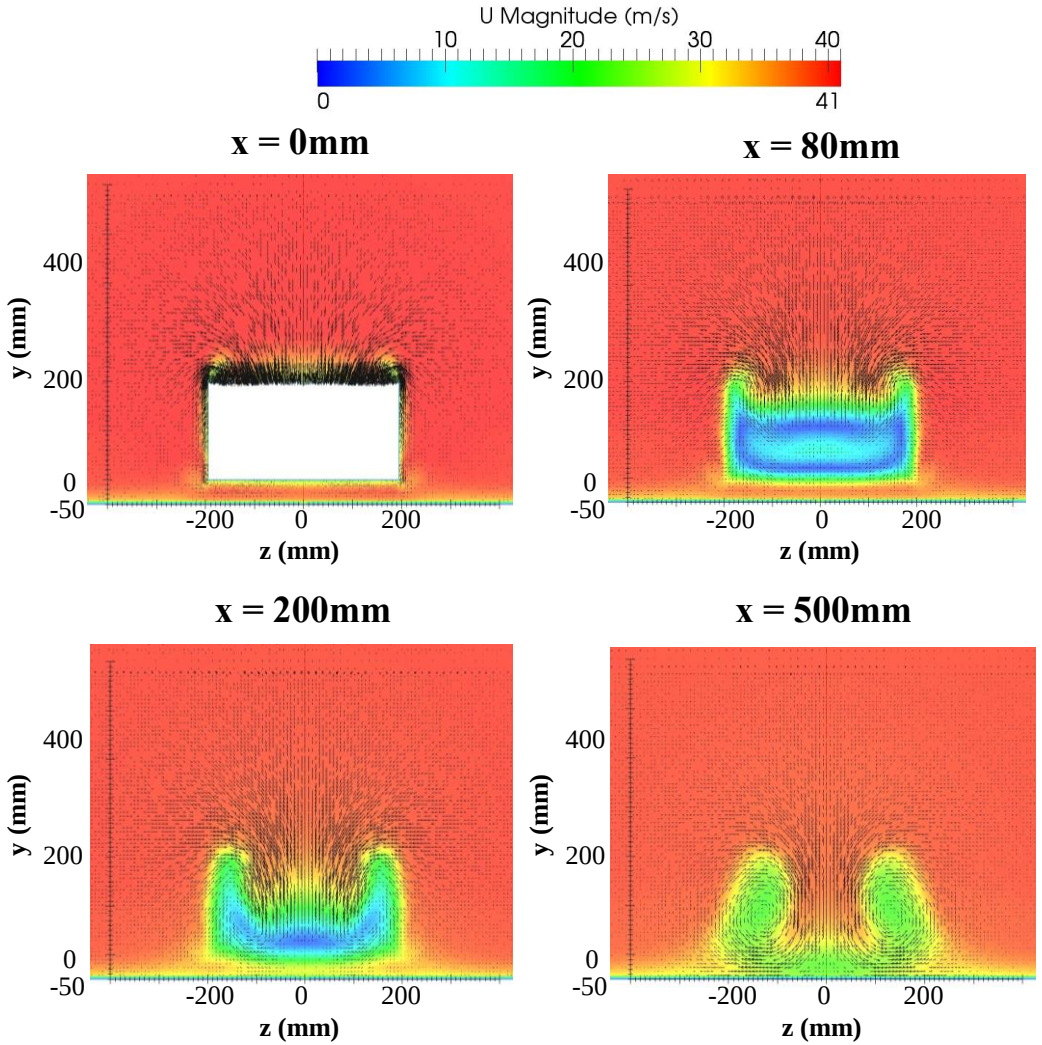


Figure 4.14 - Development of the counter-rotating trailing vortex system within the wake of the 25° slant Ahmed body, CFD model

4.2.2.3 Rear Slant Angle

CFD simulations were run for varying rear slant angles, and the results were compared with Ahmed et al. [4]. The rear slant angle was varied from 0° to 45° in 5° increments, and a rear slant angle of 12.5° was also included (as this is the point of lowest drag on the Ahmed body). The drag coefficient was plotted against the rear slant angle - see Figure 4.16. This was then compared to a similar plot by Ahmed et al. (Figure 4.15 [4]), to assess whether the CFD model was accurately predicting the drag pattern at various rear slant angles. Comparing these two figures shows that the drag coefficient at $\alpha = 0°$ appears to be quite high in comparison with drag coefficient at the other angles. Additionally separation occurs between $\alpha = 35°$ and $\alpha = 40°$ instead of at $\alpha = 30°$. Nonetheless, Figure 4.16 demonstrates that the CFD model has managed to predict the general behaviour of the drag coefficient at varying rear slant angles. In particular, the minimum drag coefficient is at $\alpha = 12.5°$, and a sudden drop in drag coefficient has been shown after the flow becomes fully separated.

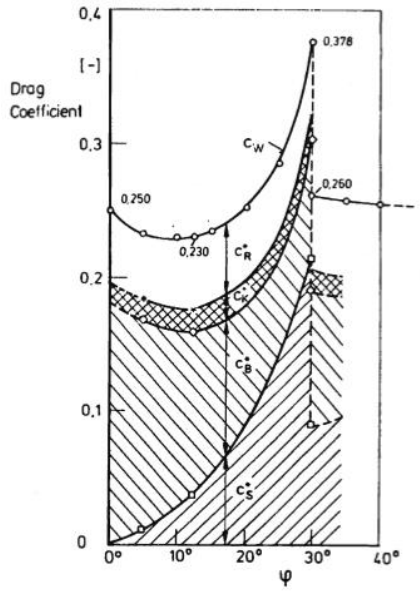


Figure 4.15 – Drag coefficient plotted against rear slant angle, Ahmed et al. [4]

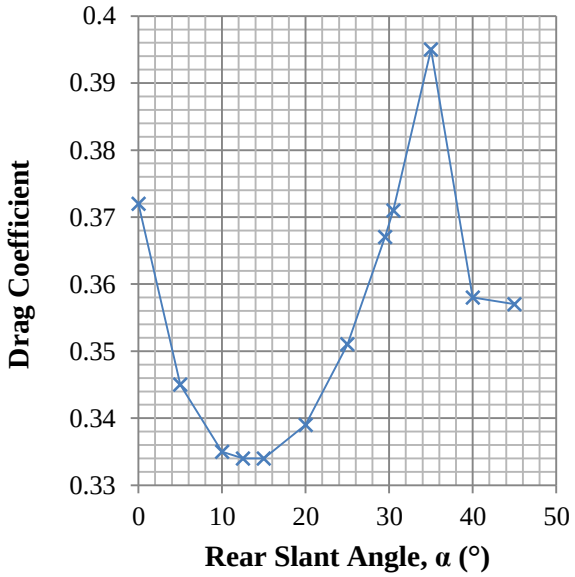


Figure 4.16 - Drag coefficient plotted against rear slant angle, CFD model

4.2.3 Pointwise Comparison

The Ahmed body was also meshed in Pointwise by Luke Hamilton [64]. Using Pointwise, Luke was able to create a good quality boundary layer mesh on the surface of the Ahmed body. Figure 4.17 and Figure 4.18 show the mesh generated along the rear slant surface (at $\alpha = 25°$) using `snappyHexMesh` and Pointwise, respectively. Figure 4.17 illustrates that the mesh generated by `snappyHexMesh` on the rear slant surface is of poor quality. In particular, the cells on the slant edge are skewed and are not parallel to the surface topology. In contrast, the mesh generated by Pointwise has created a high quality boundary layer mesh; using layers of prism elements at the wall and tetrahedral elements elsewhere. In order to make a direct comparison between the two meshing tools, the two different meshes were run under identical conditions, and the results were compared by Luke Hamilton. The first noticeable difference between the two meshes was the Pointwise mesh more accurately predicted the drag coefficient on the Ahmed body for $\alpha = 25°$ and 35° - see Table 4. Additionally, the Pointwise mesh also predicted flow separation at $\alpha = 35°$,

whereas the mesh generated by `snappyHexMesh` did not produce separated flow until α = 40°. The comparison between the two meshes demonstrated that the quality of the boundary layer mesh has a significant impact on the results - as previously thought. A more detailed comparison of the two different meshing tools can be found in Luke Hamilton's I2 report [64].

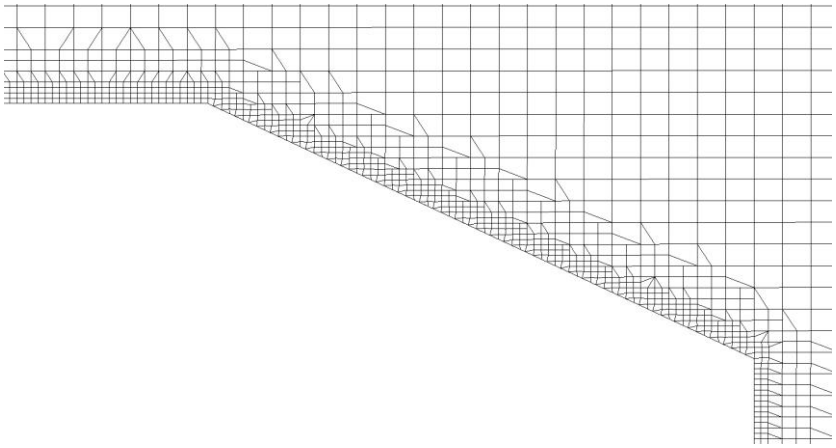


Figure 4.17 - Mesh generated on Ahmed body rear slant angle at 25° using `snappyHexMesh`

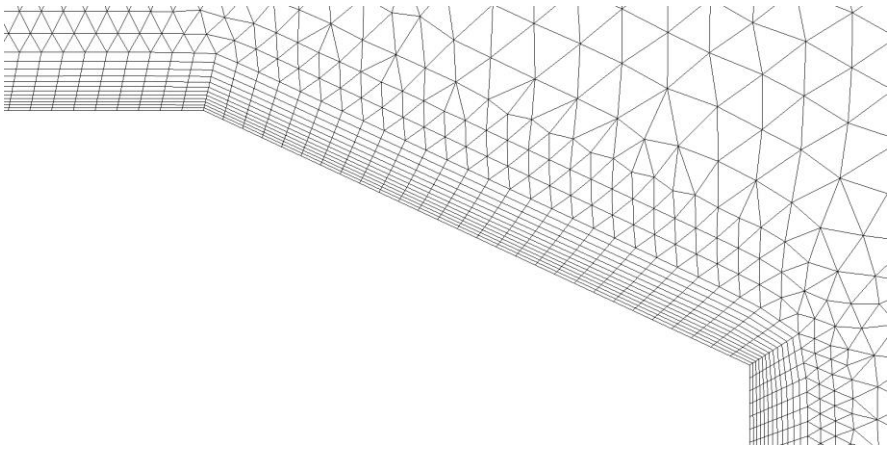


Figure 4.18 - Mesh generated on Ahmed body rear slant angle at 25° using Pointwise [64]

Table 4 – Drag coefficient comparison between `snappyHexMesh` and Pointwise

Angle (°)	Drag Coefficient				
	Meile et al. [11]	SHM	Pointwise	% Error SHM	% Error Pointwise
25	0.299	0.351	0.335	17.4	12.0
35	0.279	0.395	0.354	41.6	26.9

5 Ahmed Body Surrogate Model

5.1 CFD Automation

5.1.1 Design

After the final CFD model had been created, the next stage of the surrogate modelling process was to design an automated method to run multiple CFD simulations. This was in order to create the data to train the surrogate model. The first stage in this process was to automate the STL geometry creation in SolidWorks. It was essential to find a method that automated not only the creation of the SolidWorks part file, but also automatically exported the file in STL format. Two possible options that were already incorporated into SolidWorks were investigated. The first option was to use design tables and the second option was to use a SolidWorks add-in called DriveWorksXpress [65] (design automation software by DriveWorks). However neither of these options allows the user to automatically export in different file formats. After some research it was discovered that more powerful design automation software, DriveWorksSolo [66], was available from DriveWorks, which allows the user to export in different file formats. DriveWorksSolo is not included as standard with SolidWorks and so for this project a free trial was downloaded from the company website.

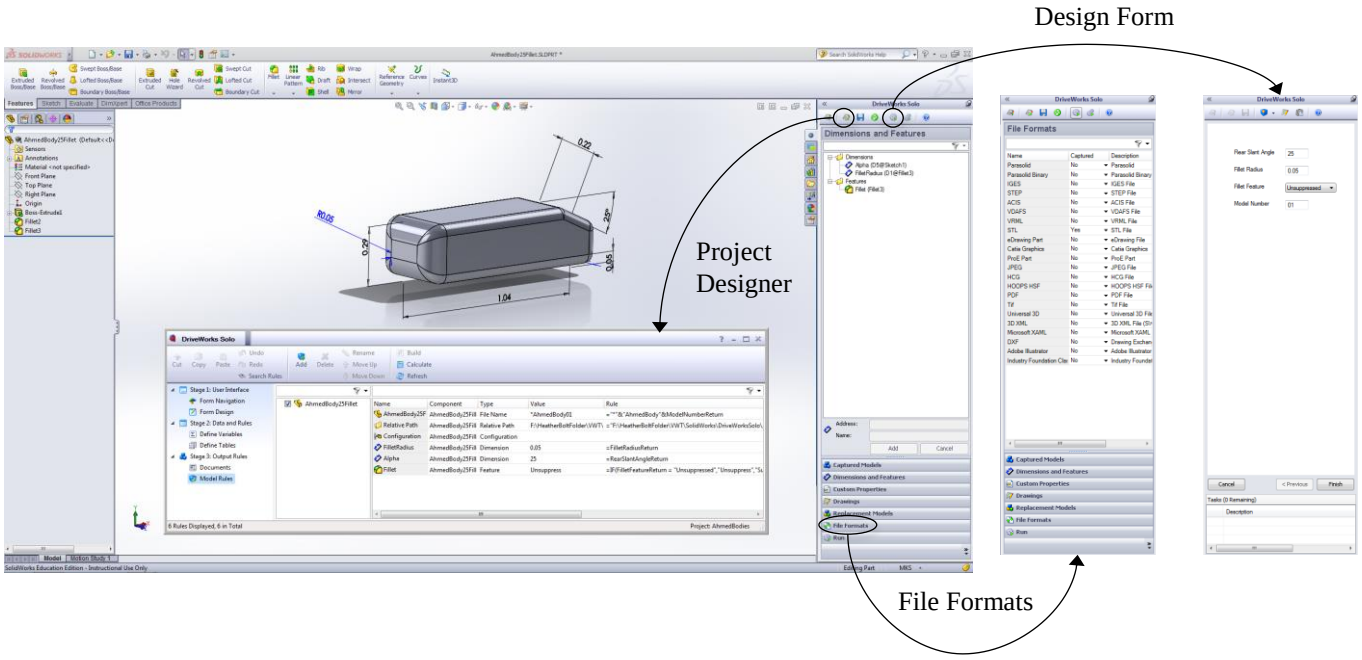


Figure 5.1 – DriveWorksSolo project of the Ahmed body in SolidWorks

A DriveWorksSolo project was created for the SolidWorks model of the Ahmed body. In this project, there were three key components: the project designer, the file formats tab, and the final design form. Figure 5.1 shows the operation of DriveWorksSolo in SolidWorks, showing these three parts. The project designer was used to capture dimensions and create rules to drive the model. Rules can be used to drive a number of parameters, including: dimension values, filenames, part drawings, file locations and feature suppression/unsuppression. For the Ahmed body, rules were created to control four different options. Firstly, the angle of the rear slant surface. Secondly, the suppression/unsuppression of the fillet feature - a fillet was applied to all sharp corners of

the Ahmed body. Thirdly, the radius of the fillet (if the fillet feature was unsuppressed) and finally, the filename. Also in the project designer, a design form was created that allowed the user to enter new data to specify and generate an Ahmed body model. The file formats tab in the project workspace enabled the Ahmed body SolidWorks model to be automatically exported in STL format. The DriveWorks project shown in Figure 5.1 was created to generate the different Ahmed body geometries used for the surrogate model training data.

In the DriveWorksSolo project, a rule was created that required the user to enter a number to name the generated Ahmed model as 'AhmedBody*Number*'. This rule was specifically designed to ensure the filenames for the STL files were a continuous string (without spaces), so they could be correctly read by a Linux operating system. The STL files were labeled sequentially from AhmedBody01 to AhmedBodyN for simplicity. Bash scripting was then used to create two script files, one script file called *Allrun*, which ran each individual CFD simulation, and a script file called *FinalScript*, which controlled the automated process. The script files for *Allrun* and *FinalScript* are in Appendix C and Appendix D, respectively. The majority of the code within these two script files was written by the author, however some code was developed with assistance from Richard Everson (Professor of Machine Learning at the University of Exeter).

The file structure for the automated set-up can be seen in Figure 5.2 below. In the main directory there is the primary script file *FinalScript* and three sub-directories: *CAD*, *Results* and *template*. The *CAD* directory contains the STL files of the Ahmed body models that were created using DriveWorksSolo. These STL files are numbered from AhmedBody01 to AhmedBodyN. The *template* directory contains the sub-directories required to run a CFD simulation. Additionally this directory also contains a script file called *Allrun* to run each individual CFD simulation and output the results. In the template folder, AhmedBodyX is used as a placeholder for the Ahmed body filename. The results from each CFD simulation are saved in the *Results* sub-directories in either: *forceCoeffs*, *forceCoeffsGraphs* or *Residuals*.

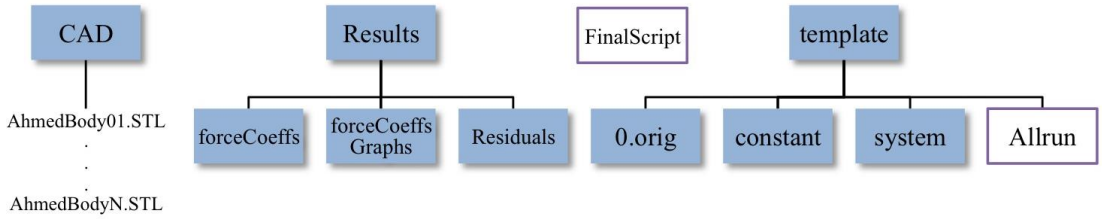


Figure 5.2 – File structure for automated simulations

Figure 5.3 shows the processes used in *Allrun* - the script file that runs each individual CFD simulation. The first step of the *Allrun* script is to create an empty file called *running*, which is deleted upon simulation completion. This file is created so the main script file (*FinalScript)* can tell whether a simulation is running. In this script the output of the following commands are saved to a file: `blockMesh`, `surfaceFeatureExtract`, `snappyHexMesh`, `potentialFoam`, and `simpleFoam`. This is so they can be viewed after a simulation has run. After `potentialFoam` has run the *`forceCoeffs`* file is replaced. The *`forceCoeffs`* file uses the `forceCoeffs` utility to output the force coefficients on the Ahmed body. This file is set to output the results for every iteration. However, if the force coefficients are set to output results for every iteration when running `potentialFoam`, this results in an error message. To avoid this error, the *`forceCoeffs`* file is set to output results for the write interval only while running

`pisoFoam`. It is then changed to output results for every time step after `pisoFoam` has run. After the simulation has run using `simpleFoam`, the utility `foamLog` is used to create a large number of residual files from the `simpleFoam` log file. Next, the graph plotting tool `xmgrace` is then used to plot graphs of the residuals and drag coefficients against iteration number and save these plots as JPEG files. These pictures are copied across to the *Results* directory, in addition to the data file for the force coefficients. This places the results for all the simulations in one place – making it quick and easy to assess the results.

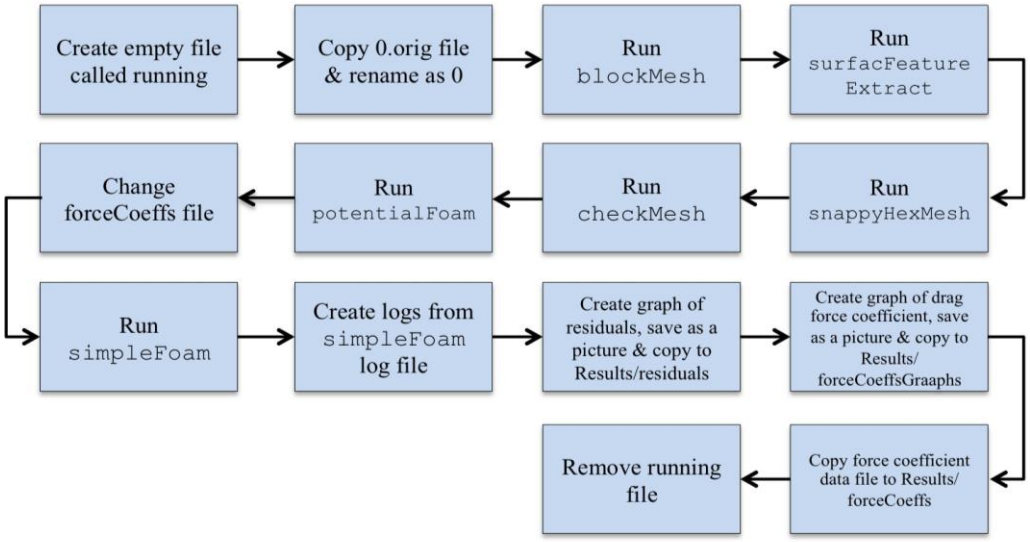


Figure 5.3 – Processes involved in Allrun script file

Figure 5.4 overleaf is a flowchart representation of the *FinalScript* script file - this file controls the overall automated process. The first operation of *FinalScript* is to rename the Ahmed body geometry files from AhmedBodyN.STL to AhmedBodyN.stl. This step is required because SolidWorks saves the geometry files as filename.STL whereas in OpenFOAM the required format is filename.stl. The next step is to read the number of jobs to run in parallel – *N*. This controls how many simulations are run in parallel and can be easily changed by the user. The script file then lists the names of all the files in the CAD directory, and checks to see if there is a directory for each filename in the main directory. If there is not, the script then counts the number of simulations that are running, if this number is less than N (the number of jobs to run in parallel) it runs the next model in the numeric sequence (starting at AhmedBody01). For illustrative purposes, assume the next model in the sequence is AhmedBody06. To run this model, first the template directory is copied and renamed as *AhmedBody06*. The geometry, AhmedBody06.stl, is then copied from the CAD directory and pasted into *AhmedBody06/constant/triSurface*. Afterwards the *AhmedBody06* directory is searched and every occurrence of AhmedBodyX is renamed to AhmedBody06. Following this, the *Allrun* script is run in the background. Then, it checks: how many jobs there are in total (by counting the number of files in the CAD directory); how many jobs have been started (by counting the number of AhmedBodyX model directories); and how many jobs are running (by counting the number of *running* files in the AhmedBodyX model directories). This information is printed to screen. If the number of jobs running equals 0 and the number of jobs started equals the total number of jobs, the script compiles the last line of every data file in *Results/forceCoeffs* directory to produce one file with the final drag coefficients for all the results. Otherwise, it loops back to the stage where it checks to see if there is a directory for each AhmedBodyX model.

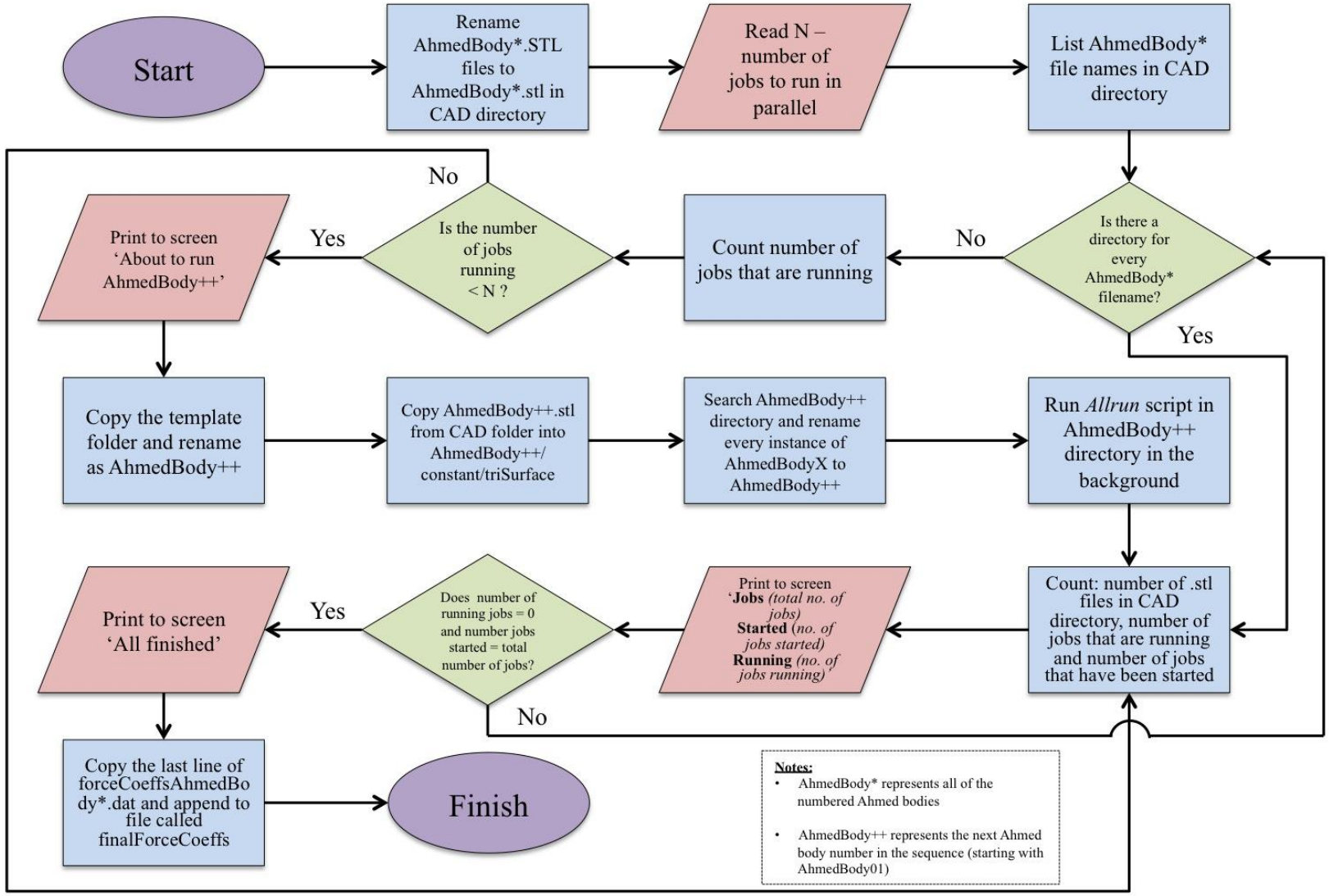


Figure 5.4 – A flowchart representation of the FinalScript file

5.1.2 Design of Experiment

The Design of Experiment (DoE) is 'the sampling plan in the design variable space' [43]; it is how the training data for the surrogate model is chosen. The DoE requires careful consideration, it must be 'constructed so as to give an even spread of data throughout the domain of interest' [67]. Insufficient data will mean the surrogate model fails to train properly, and may lead to erroneous results. On the other hand generating too much data is a waste of computational resources. Key points to consider include: the number of input and output parameters; the number of simulations that can be run with the computational resources available; and how the data is shaped within the design space. As the number of input and output parameters increases so does the complexity of the design variable space, thus the required number of data points to sufficiently train the model also increases. Consequently, it is important to consider the available computational resources when creating a surrogate model. The design variable space can be sampled randomly, however it is more useful to consider how the data is shaped within the design space and position more sampling points in areas of interest (if possible).

It was decided to create a surrogate model for the Ahmed body that predicted one output parameter - the drag coefficient – based on two input parameters. The first input parameter was the angle of the rear slant surface. The second input parameter was a fillet radius on the sharp edges of the Ahmed body. Seven rear slant angles were chosen to capture the pattern of the drag coefficient at: 5°, 12.5°, 20°, 25°, 35°, 40° and 45°. Five different fillet radii were also chosen at: 0mm, 7.5mm, 15m, 30mm and 50mm. These points were deliberately clustered to include more points at smaller radii. This clustering was influenced by an investigation by Dan Nima [68]. In his investigation, Dan found that filleting the sharp edges of a car-type bluff body reduced the drag coefficient, however the rate of change of drag decreased with increasing fillet radius (see Figure 5.5 [68]). Based on this information, more data points were chosen at lower fillet radii. Figure 5.6 shows the DoE for the Ahmed body surrogate model, displaying all thirty-five data points that were chosen.

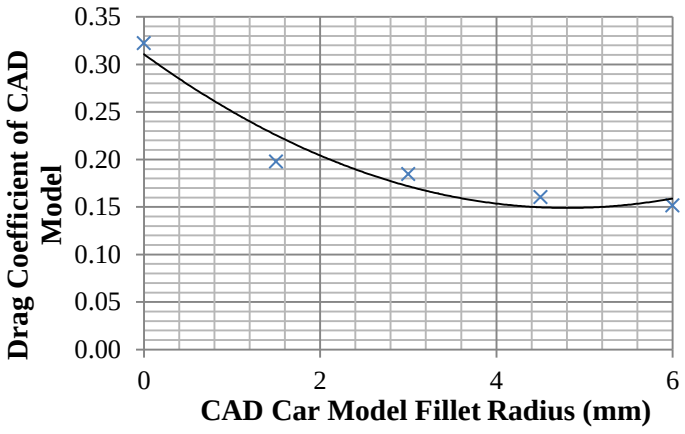


Figure 5.5 – Graph of drag coefficient plotted against fillet radii for basic car model [68]

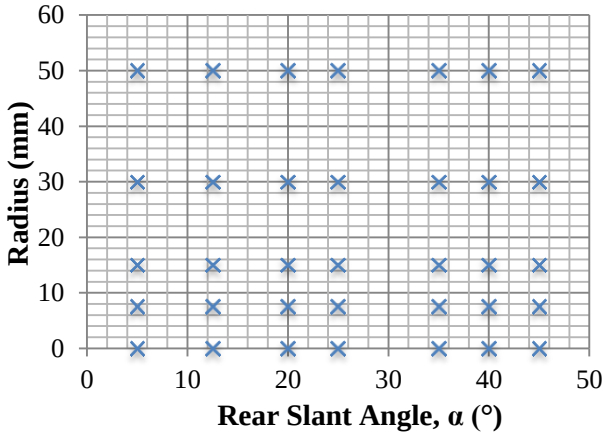


Figure 5.6 – Design of Experiment for Ahmed body surrogate model

5.1.3 Results

The thirty-five different Ahmed body simulations were run using the automated process as described in section 5.1.1. The total time to mesh and run these simulations was 55 hours when running in parallel on 12 cores @ 3.47GHz. Figure 5.7 shows the results from the simulations. In the *Results* directory the *forceCoeffsGraphs*, *Residuals* and *forceCoeffs* sub-directories are populated with the results from the simulations. Images of the residuals and drag coefficients can be easily viewed to quickly spot any simulations that may not have converged. The final output of the automated process is a data file containing: the names of the simulations, the number of iterations the simulations ran for and the final drag coefficient.

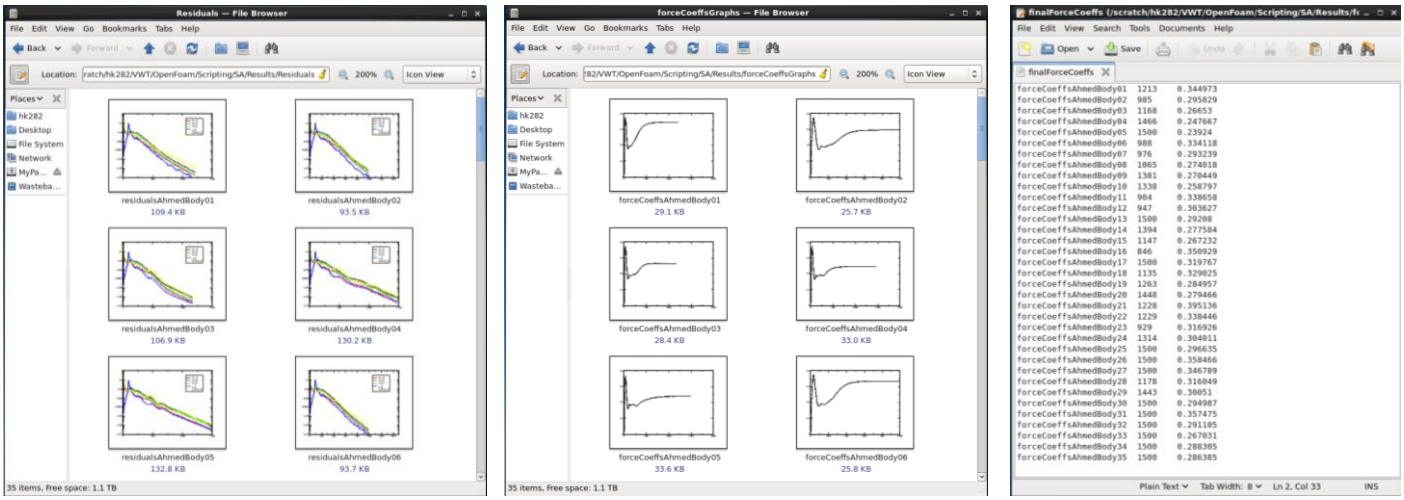


Figure 5.7 - Results from the automated CFD simulations showing: residual graphs (left image) drag coefficient graphs (middle image) and compiled data file containing the final drag coefficients for all simulations (right image)

5.2 Artificial Neural Network

5.2.1 Design

A comma-separated values (CSV) file was created with the two input parameters and corresponding drag coefficient for each simulation. This data was imported into Matlab and plotted as a surface, see Figure 5.8. This figure shows that overall the drag coefficient has behaved as expected. The areas of lowest drag coefficient are at high fillet radii and the drag coefficient peaks at $\alpha = 35°$ (with no fillet) – this corresponds to the peak shown in Figure 4.16.

This data was then used to train a type Artificial Neural Network (ANN) known as a multi-layered perceptron (MLP). The MLP was created and optimised in Matlab [69] (a commercially available computing language and environment) by using the Netlab [70] toolkit. This toolkit contains Matlab functions and scripts for the creation of neural networks and related models. An MLP network was chosen as it is 'probably the most widely used architecture for practical applications of Neural Networks' [70]. As discussed in the methodology section, due to project delays Professor Richard Everson created the MLP described in this report. Weight decay regularisation was used to optimise the training of the MLP. Prof. Everson created two Matlab functions – *trymlp.m* and *loocv.m*, which can be found in Appendix E and Appendix F, respectively. The *trymlp.m* function standardised the training data and initialised the MLP network several times to improve

the performance of the network. The *loocv.m* function used LOOCV to assess the test error and tried a range of different λ values. The MLP was trained using the best from three initialisations and with ten values of λ varied logarithmically from 1×10^{-8} to 10; the results can be seen in Figure 5.9.

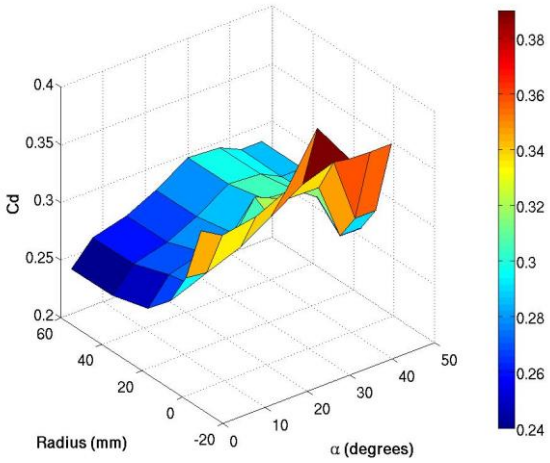


Figure 5.8 – Surface plot of drag coefficient against fillet radius and rear slant angle (training data)

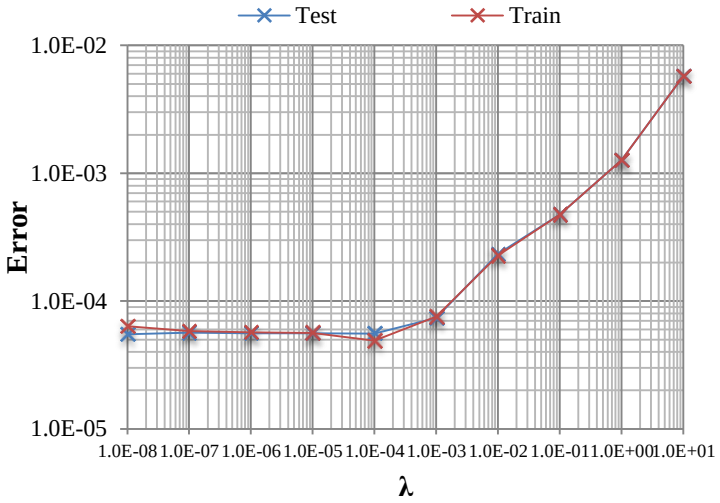


Figure 5.9 – Test error and training error plotted against the regularisation coefficient

5.2.2 Results & Analysis

Figure 5.9 shows that as the value of λ (the regularisation coefficient) decreases so does the test and training error, until $\lambda = 1\times10^{-4}$. After this point the test error increases slightly. This increase in test error is due to overfitting, however this increase is negligible. This indicates that the parameter space has been sufficiently well defined by the training data to prevent any significant overfitting. The test and training error for the MLP at $\lambda = 1\times10^{-4}$ was 4.9×10^{-5} and 5.6×10^{-5}, respectively. These are the squared errors and so correspond to a percentage test and training error of 2.3% and 2.5%, respectively. After the MLP had been trained, 600 data points (i.e. a fillet radius and a rear slant angle) were randomly generated, this information was provided to the trained MLP and the drag coefficient was predicted for these points – the results are shown in Figure 5.10. It took just under three minutes to train and optimise the MLP and then predict the drag coefficient for the 600 randomly generated points. However, it would have taken around 39 days to run the equivalent 600 points as CFD simulations under the same conditions that were used to generate the training data (12 cores running in parallel @ 3.47 GHz). These results have shown that for two input parameters, a surrogate model has been able to accurately predict the drag coefficient for the Ahmed body. These results have also shown that the Ahmed body surrogate model is significantly computationally cheaper than the Ahmed body CFD model.

One of the primary reasons for the success of the surrogate model is that the parameter space has been adequately defined by the training data. As only two input parameters were chosen, the 35 points were sufficient to describe the behaviour of the Ahmed body CFD model. However, as the number of parameters increases so does the complexity of the model and therefore the number of points needed to sufficiently describe the design space

also increases. If more input parameters were to be investigated, significantly more data points would be needed to sufficiently train the ANN.

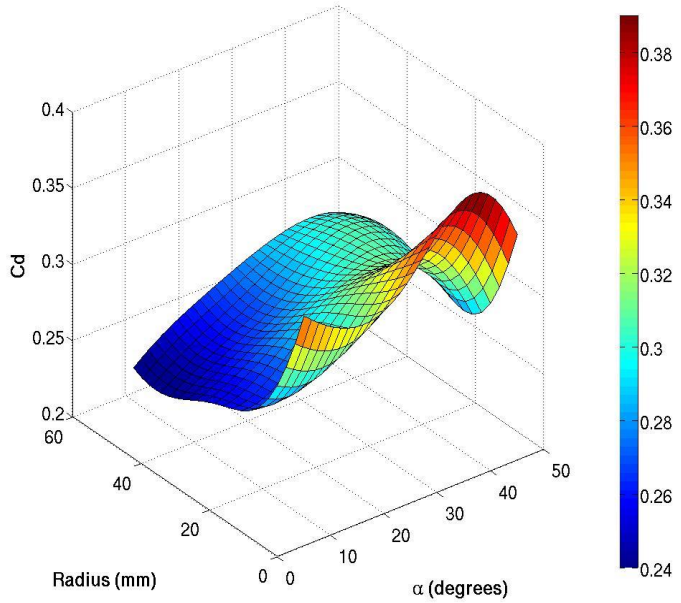


Figure 5.10 - Surface plot of drag coefficient plotted against fillet radius and rear slant angle (MLP results)

The Ahmed body surrogate model was able to accurately predict the drag coefficient of the Ahmed body CFD model. However, as demonstrated in section 4.2.2, this CFD model predicted the general behaviour of the drag coefficient but failed to accurately predict the value of the drag coefficient. The Ahmed body CFD model had errors of 17.4% and 41.6%, for $\alpha = 25°$ and 35°, respectively. In the work undertaken by Luke Hamilton on a more accurate Ahmed body CFD model, he created a model that had errors of -2.3% and 5.4%, for $\alpha = 25°$ and 35°, respectively. For these Ahmed body CFD models, Luke used a structured mesh of 2.2 million elements, the S-A RANS turbulence model, and 2nd order differencing - see [64] for more details. These simulations took approximately 13 hours each to run in parallel on 8 cores @ 3.47 GHz. To re-run the 35 Ahmed body simulations using Luke's more accurate CFD model, it would take approximately 13 days (running in parallel on 12 cores @ 3.47 GHz).

The purpose of the Ahmed body surrogate model was to investigate the use of a surrogate-based modelling approach in automotive design. The surrogate model of the Ahmed body created in this project has been highly successful, predicting the drag coefficient of the Ahmed body with an average error of 2.3% - whilst being vastly quicker than the Ahmed body CFD model. This indicates that automotive design optimisation studies may be possible using a surrogate model, where it would have been impossible otherwise. In this project, the surrogate model was limited by the computational resources available. Consequently, only two geometric parameters were considered as the input data and the CFD model had to compromise on accuracy in order to limit its computational expense. Nonetheless, where CFD is used in the automotive industry, the computational resources available are considerably greater than what was obtainable in this project. With more computational resources, a more accurate, more complex surrogate model could be created, which has the potential to be a powerful tool in the automotive design industry.

6 Sustainability

Reducing the drag coefficient is a major consideration for external automotive design because of the direct link between aerodynamic drag and fuel consumption. Early car design was not concerned with external aerodynamics, however the oil crisis in the 1970s focused the design attention on fuel consumption and drag reduction. Today, with rising fuel prices and a greater awareness of environmental issues, customer demand is driving the need to lower fuels consumption in car design. Figure 6.1 [71] highlights the energy consumption of a car. At lower speeds (i.e. driving in an urban environment) the fuel consumption caused by aerodynamic drag is low in comparison to the rolling resistance of the car wheels. On the other hand, at higher speeds (i.e. motorway driving) the air resistance on the car is the significant contribution to fuel consumption.

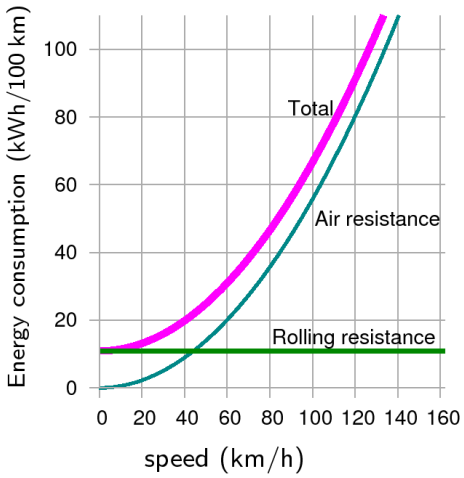


Figure 6.1 – Simple theory of car fuel consumption (energy per distance) when driving at steady speed. Assumptions: the car's engine uses energy with an efficiency of 0.25, whatever the speed; $c_d A_{car} = 1m^2$; $m_{car} = 1000kg$; and $C_{rr} = 0.01$ [71]

CFD is used to examine the aerodynamic properties of a car. As shown in this report, the flow around even a simple car model is very complex. Thus CFD based optimisation for automotive aerodynamics is often unattainable because of the significant computational expense involved. Surrogate-based optimisation is a possible solution to this problem. This report has shown that a surrogate model has been successful in predicting the drag coefficient of a relatively inexpensive CFD model of the Ahmed body. If surrogate-based optimisation was implemented in the automotive industry it could have a significant impact on design optimisation. With 2.87 million cars on the roads in Great Britain alone [72], any small decrease in drag coefficient would drastically cut fuel consumption; reducing energy consumption, CO_2 emissions and pollution.

7 Conclusions

In conclusion, this project has created a CFD model of the Ahmed body in OpenFOAM, and investigated the use of a surrogate-based modelling approach for the Ahmed body. The CFD model was validated against experimental literature of the Ahmed body by comparing: the drag coefficient values, the wake flow, and the relationship between the rear slant angle and the drag coefficient. It was essential that this CFD model had the right balance between accuracy and computational expense in order to generate the quality and quantity of data needed to train the surrogate model. Due to this compromise, the CFD model showed poor agreement with experimental drag coefficient values. It was also found that the `snappyHexMesh` utility in OpenFOAM was unable to generate an adequate boundary layer mesh on the surface of the Ahmed body. This also affected the accuracy of the CFD model results. However, the Ahmed body CFD model was able to predict the general relationship between the drag coefficient and the rear slant angle, as well as key features of flow in the wake region.

The creation of model geometry, the meshing process, and the solution output was automated using DriveWorkSolo in SolidWorks, and bash scripting. This was found to be a highly successful way to automatically run multiple simulations. Thirty-five different Ahmed body models were created, varying the rear slant angle and a fillet radius of the sharp edges, and run in OpenFOAM. An artificial neural network was created (by Prof. Richard Everson) and trained on the data generated by these CFD simulations. This surrogate model was able to predict the drag coefficient based on the two geometric input parameters with an average error of 2.3%.

This project has demonstrated that a surrogate-based modelling approach has been successful in predicting the drag coefficient for an Ahmed body CFD model, based on two geometric parameters. The Ahmed body surrogate model was significantly less computationally expensive than the Ahmed body CFD model. Thus indicating that automotive design optimisation studies may be possible using a surrogate model where it would have been impossible otherwise. Surrogate modelling has the potential to be a powerful tool for automotive design. With more time and computational resources, future work could be conducted into the use of more accurate CFD models and more complex surrogate models (i.e. using more parameters).

8 References

1. OpenFOAM Foundation, OpenFOAM (Version 2.2.2) [Open Source CFD Code] Available at: http://www.openfoam.com/ [Accessed 17/04/2014].

2. Bolt, H., 2013. *Individual Report 1: Virtual Wind Tunnel Project*. MEng Report. University of Exeter.

3. CFD Online, 2005. *File:Ahmed.gif.* [online] Available at: http://www.cfd-online.com/Wiki/File:Ahmed.gif [Accessed 28/09/2013].

4. Ahmed, S. R., Ramm, G., Faltin, G., 1984. *Some salient features of the time averaged ground vehicle wake.* SAE Technical Paper No. 840300.

5. Hinterberger, C., García-Villalba, Rodi, W., 2004. *Large eddy simulation of flow around the Ahmed Body.* In: The Aerodynamics of Heavy Vehicles: Trucks, Buses, and Trains, Volume 1. Berlin; Heidelberg; New York: Springer.

6. Douglas, J. F., 2005. *Fluid Mechanics.* 5th ed. Harlow: Pearson/ Prentice Hall. Chapter 11: Boundary Layer.

7. Hucho, W., 1993. Aerodynamics of Road Vehicles. *Annual Review of Fluid Mechanics.* **25**(1993), pp. 485 – 537.

8. Franck, G., Nigro, N., Storti, M., D'Elía, J., 2009. Numerical Simulation of the Flow Around the Ahmed Vehicle Model. *Latin American Applied Research.* **39**(4).

9. Bayraktar, I., Landman, D., and Baysal, O., 2001. *Experimental and Computational Investigation of Ahmed Body for Ground Vehicle Aerodynamics.* SAE Technical Paper 2001-01-2742.

10. Spohn, A., & Gillieron, P., 2002. Flow separations generated by a simplified geometry of an automotive vehicle. In: IUTAM Symposium*: Unsteady Separated Flows.* Toulouse, France, 2002.

11. Meile, W., Brenn, G., Reppenhagen, A., Lechner, B., Fuchs, A., 2011. Experiments and numerical simulations on the aerodynamics of the Ahmed body. *CFD Letters.* **3** (1): 32 – 39.

12. Sims-Williams, D., and Duncan, B., 2003 *The Ahmed Model Unsteady Wake: Experimental and Computational Analyses.* SAE Technical Paper 2003-01-1315.

13. Lienhart, H., Soots, C., and Becker, S., 2003. *Flow and turbulence structures in the wake of a simplified car model (Ahmed Model).* SAE Technical Paper 2003-01-0656.

14. ERCOFTAC, 2001. *10th joint ERCOFTAC (SIG-15) -IAHR-QNET/CFD Workshop on Refined Turbulence Modelling: CASE 9.4 Flow around a simplified car body (Ahmed body).* Available at: http://www.ercoftac.org/fileadmin/user_upload/bigfiles/sig15/database/9.4/ws_10.html [Accessed: 17/04/2014].

15. Chauhan, R. B., Thundil, K. R. R., 2012. Numerical Investigation of External Flow around the Ahmed Reference Body Using Computational Fluid Dynamics. *Research Journal of Recent Sciences.* **1**(9), pp. 1-5.

16. Krajnovic, S., Davidson, L. 2004. *Large-Eddy Simulation of the Flow Around Simplified Car Model.* SAE Paper No. 2004-01-0227.

17. Clancy, L. J., 1975. *Aerodynamics.* London: Pitman Publishing. Chapter 4: Fundamentals of Air Flow

18. Versteeg, H.K., and Malalalsekera, W., 2006. *An introduction to computational fluid dynamics: the finite volume method.* 2nd ed. Harlow: Pearson/ Prentice Hall. Chapter 2: Conservation Laws of Fluid Motion.

19. Versteeg, H.K., and Malalalsekera, W., 2006. *An introduction to computational fluid dynamics: the finite volume method.* 2nd ed. Harlow: Pearson/ Prentice Hall. Chapter 1: Introduction.

20. OpenFOAM Foundation, 2011. *Numerical Method* [online] Available at: http://www.openfoam.org/features/numerical-method.php [Accessed 15/04/2014].

21. OpenFOAM Foundation, 2013. *OpenFOAM User Guide (Version 2. 2. 2).* [pdf] Available at: www.openfoam.org/docs/user. [Accessed 01/10/2013] Chapter 1: Introduction.

22. Minguez, M., Pasquetti, R., Serre, E., 2008. High-order large-eddy simulation of flow over the "Ahmed body" car model. *Physics of Fluids.* **20**(9), pp. 095101: 1 – 17.

23. Franck, G., D'Elía, J., 2004. CFD modeling of the flow around the Ahmed vehicle model. In: *The Second Conference on Advances and Applications of GiD*, February 18–20, 2004, Barcelona, Spain, Paper No: 216.

24. Brondolo, L., 2011. *Comparative Investigation of Large Eddy Simulation and RANS Approaches for External Automotive Flows.* MSc. Cranfield University.

25. Serre, E., Minguez, M., Pasquetti, R., Guilmineau, E., Deng, G. B., Kornhaas, M., Schäfer, M., Fröhlich, J., Hinterberger, C., Rodi, W., 2013. On simulating the turbulent flow around the Ahmed body: A French–German collaborative evaluation of LES and DES. *Computers and Fluids.* **78**, pp. 10 – 23.

26. Kapadia, S., Roy, S., Vallero, M., Wurtzler, K., Forsythe, J., 2003. Detached-Eddy Simulation over a Reference Ahmed Car Model. In: ERCOFTAC, *Proceedings of*

the fifth international ERCOFTAC Workshop on direct and large-eddy simulation. Munich, 27th – 29th August 2003. Netherlands: Springer.

27. Ahmad, N. E., Abo-Serie, E., Gaylard, A., 2010. Mesh Optimisation for Ground Vehicle Aerodynamics. *CFD Letters*, **2**(1), p. 54 – 65.

28. Casey, M. and Wintergerste, T. (Eds.), 2000. *Best Practices Guidelines (Version 1.0)*, ERCOFTAC (European Research Community on Flow, Turbulence and Combustion) Special Interest Group on Quality and Trust in Industrial CFD.

29. CFD Online, 2011. *Dimensionless Wall Distance.* [online] Available at: http://www.cfd-online.com/Wiki/Dimensionless_wall_distance_(y_plus) [Accessed 10/11/2013].

30. Tabor, G., 2013. *Fluid Mechanics Lecture 5,* Thermofluids and Energy Conversion, ECM3151. [online via internal ELE] The University of Exeter. Available at: http://vle.exeter.ac.uk/course/view.php?id=747 [Accessed 25/03/2014].

31. CFD Online, 2008. *Wall Functions*. [online] Available at: http://www.cfd-online.com/Wiki/Wall_Functions [Accessed: 25/03/2014].

32. CFD Online, 2011. *Law of the Wall.* [online] Available at: http://www.cfd-online.com/Wiki/Law_of_the_wall [Accessed: 25/03/2014].

33. Minguez, M., Pasquetti, R., Serre, E., 2008. High-order large-eddy simulation of flow over the "Ahmed body" car model. *Physics of Fluids.* **20**(9), pp. 095101: 1 – 17.

34. Frei, W., 2013. *Which Turbulence Model Should I Choose for my CFD Application?* [online] Available at: http://www.comsol.com/blogs/which-turbulence-model-should-choose-cfd-application/ [Accessed 15/11/2013].

35. Leary, S. J., Bhaskar, A., Keanea, A. J., 2004. A Derivative Based Surrogate Model for Approximating and Optimizing the Output of an Expensive Computer Simulation. *Journal of Global Optimization*. **30**, pp. 39–58.

36. Ghent University, 2014. *SUMO – Surrogate Modelling Lab | Surrogate Models* [online] Available at: http://www.sumo.intec.ugent.be/surrogates [Accessed 10/02/2014].

37. Asoka, S. P., Sankaranarayanasamyb, K., Sundararajanc, T., Rajeshd, K., Ganeshane, G. S., 2007. Neural network and CFD-based optimisation of square cavity and curved cavity static labyrinth seals. *Tribology International.* 40, pp.1204–1216.

38. Shang, Z., 2005. Application of artificial intelligence CFD based on neural network in vapor–water two-phase flow. *Engineering Applications of Artificial Intelligence.* **18**, pp. 663–671.

39. Kuan, Y., Hsueh, Y., Lien, H., Chen, W., 2006. Integrating Computational Fluid Dynamics and Neural Networks to Predict Temperature Distribution of the Semiconductor Chip with Multi-heat Sources. In: Advances in Neural Networks, *Third International Symposium on Neural Networks.* Chengdu, China: May/ June 2006, pp. 1005 – 1013.

40. Kalogirou, S. A., 2001. *Artificial neural networks in renewable energy systems applications: a review.* Renewable and Sustainable Energy Reviews **5**(4): 373-401.

41. Gershenson, C., 2013, *Artificial Neural Networks for Beginners.* [online] Available at: http://arxiv.org/ftp/cs/papers/0308/0308031.pdf [Accessed 11/11/2013].

42. Bishop, C. M., 2006. *Pattern Recognition and Machine Learning.* New York: Springer .

43. Queipo, N. V., Haftka, R. T., Shyy, W., Gol, T., Vaidyanathan, R., Tucker, P. K., 2005. Surrogate- based analysis and optimization. *Progress in Aerospace Sciences.* **41,** pp. 1-28.

44. Word Press, 2013, *The Shape of Data: General Regression and Over Fitting.* [online] Available at: http://shapeofdata.wordpress.com/2013/03/26/general-regression-and-over-fitting/

45. ALGIB, 2014. *Neural Networks: Improving Generalisation* [online] Available at: http://www.alglib.net/dataanalysis/improvinggeneralization.php [Accessed 20/03/2014]

46. Pointwise Inc. 2013 *Pointwise*. (Version 17) [Mesh Generation Software for CFD]

47. Kitware, ParaView (Version 3.12.0) [Open-source Visualisation Software]

48. OpenFOAM Foundation, 2013. *User Guide (Version 2.2.2).* [pdf] Available at: www.openfoam.org/docs/user. Chapter 5: Mesh Generation and Conversion. Section 5.4: Mesh generation with the snappyHexMesh utility.

49. Dassault Systèmes, Solidworks (Version 2014) [3D CAD Design Software].

50. OpenFOAM Foundation, 2013. *User Guide (Version 2.2.2).* [pdf] Available at: www.openfoam.org/docs/user. Chapter 5: Mesh Generation and Conversion. Section 5.5: Mesh Conversion.

51. Newham, C., Rosenblatt, B., 2005. *Learning the bash Shell: Unix Shell Programming*.3rd ed. Sebastopol: O'Reilly Media, Inc.

52. Blades, L., 2013. *Individual Report 1: Virtual Wind Tunnel Project*. MEng Report. University of Exeter.

53. Docherty, D., 2013. *Individual Report 1: Virtual Wind Tunnel Project*. MEng Report. University of Exeter.

54. University of Exeter, 2013. *Setting up your workstation.* [pdf] Available at: https://www.exeter.ac.uk/media/universityofexeter/humanresources/documents/healthsafety/dse/Setting_up_your_workstation_-_Jan_2013.pdf [Accessed 01/10/2013]

55. University of Exeter, 2013. *Health and Safety Standard: Display Screen Equipment and Portable Workstations.* [online] Available at: https://www.exeter.ac.uk/staff/wellbeing/safety/guidance/dse/

56. CFD Online, 2014. *Turbulence free-stream boundary conditions.* Available at: http://www.cfd-online.com/Wiki/Turbulence_free-stream_boundary_conditions [Accessed 25/03/2014].

57. Rumsey, C., 2013. *NASA Turbulence Modelling Resource | The Spalart-Allmaras Turbulence Model.* [online] Available at: http://turbmodels.larc.nasa.gov/spalart.html [Accessed 25/03/2014].

58. OpenFOAM Foundation, 2013. *OpenFoam User Guide (Version 2.2.2)* [pdf] Available at: www.openfoam.org/docs/user. Chapter 7: Models and Physical Properties. Section 7.2: Turbulence Models.

59. OpenFOAM Foundation, 2013. *OpenFoam User Guide (Version 2.2.2)* [pdf] Available at: www.openfoam.org/docs/user. Chapter 4: OpenFOAM Cases, Section 4.4: Numerical Schemes

60. Tu, J., Yeoh, G. H., Liu C., 2008. *Computational Fluid Dynamics: A Practical Approach.* Amsterdam: Butterworth-Heinemann.

61. OpenFOAM Foundation, 2014. *Run-time Post-Processing.* [online] Available at: http://www.openfoam.org/features/runtime-postprocessing.php [Accessed 20/03/2014]

62. Spalart, P. R. Allmaras, S. R., 1994. A One-Equation Turbulence Model for Aerodynamic Flows. *Recherche Aerospatiale*. **1**, pp. 5-21

63. NASA 2008. *Overview of CFD Verification and Validation.* [online] Available at: http://www.grc.nasa.gov/WWW/wind/valid/tutorial/overview.html [Accessed 15/02/2013

64. Hamilton. L., 2014. *Individual Report I2: Virtual Wind Tunnel Project.* MEng Report. University of Exeter.

65. DriveWorks Ltd, DriveWorks Express, 2014 [Design Automation Software for use in SolidWorks].

66. DriveWorks Ltd, DriveWorks Solo, 2014 [Design Automation Software for use in SolidWorks].

67. Forrester, A., Bressloff, N., Keane, A., 2006. Optimization using surrogate models

and partially converged computational fluid dynamics simulations. *Proceedings of the Royal Society A.* **462**, pp. 2177–2204

68. Nima. D., 2014. *Individual Report I2: Virtual Wind Tunnel Project.* MEng Report. University of Exeter.

69. Mathworks, Matlab (R2012a) [Computer program].

70. Nabney, I., Bishop, C., Netlab [Matlab functions and scripts].

71. Mackay, D. J. C., 2009. *Sustainable Energy: Without the Hot Air*. Cambridge: UIT Cambridge Limited.

72. National Statistics, 2012. *Vehicle Licensing Statistics Q2 2012.* [online] Available at : https://www.gov.uk/government/publications/vehicle-licensing-statistics-q2-2012 [Accessed 20/04/2014].

Dummy Page

9 Appendices

Appendix A. DSE User Self Assessment Form [54]

To be completed by the *DSE user* after implementing the guidance given in the "Guidance for setting up your workstation safely".

Answer all of the questions, and for those with ***NO*** as your answer, please add the points together giving you a total score at the end of the assessment. Use your total score to determine if any action needs to be taken.

If you needed to mark any questions with an asterisk (*) please contact your manager despite the score who will assess and make any necessary referrals to the Safety Service.

Name of DSE User **Heather Bolt** **College / Service / Dept**

Date of Assessment **01/10/2013** **CEMPS**

DSE Component		NO	Action Required / Comments
Desk			
Is there enough space on your desk top for the flow of work?	**1**	**1**	
Have you got enough leg room?	**2**	**Y**	
Is the desk deep enough for you to have the monitor set between 450mm and 650mm from your eyes, when you are seated in the correct position?	**2***	**Y**	
Is there enough room for a space between your keyboard and you for your wrists to rest on the desk (4-6 inches / 10-14cm recommended) between typing	**2***	**Y**	
Is your desk surface free from reflection?	**1**	**Y**	
Chair			
Is your chair at a height where the bottom of your elbows are at the same height as the keyboard when using the keyboard?	**2**	**Y**	
Does the back rest support the small of your back in an upright posture?	**1**	**1**	
Can you sit back into the chair seat fully without any pressure behind the knees?	**2**	**Y**	
If fitted, are armrests set up correctly i.e not preventing adequate movement of the chair?	**1**	**Y**	
Can you get close to the desk to type with the elbows vertically under the shoulders?	**2**	**Y**	
Is the chair comfortable?	**1**	**Y**	
Is the chair stable and all adjustment levers working?	**2**	**Y**	

DSE Component		NO	Action Required / Comments
With seat height adjusted correctly for the elbows, can you place your feet firmly on the floor without compressing the underside of your thighs?	**2**	**Y**	
If a footrest is required, have you got access to one?	**2**	**N/A**	No footrest
Monitor			
Is the monitor / screen between 450mm-650mm away from your eyes (arms length)	**2**	**Y**	
Is the monitor directly in front of you?	**2**	**Y**	
Are your eyes level with the top of the screen?	**1**	**Y**	
Is the screen free from glare / reflections?	**2**	**Y**	
Is the information on the screen well defined and easy to read?	**1**	**Y**	
The image is flicker free?	**2**	**Y**	
Do you clean the screen regularly?	**1**	**Y**	
Is the monitor tilted between 5 and 15% off the vertical?	**1**	**Y**	
Can you adjust the brightness and contrast easily?	**1**	**Y**	
Keyboard			
Is the keyboard at the correct angle to prevent any up or down bending of the wrist?	**2**	**2**	
Is your keyboard far enough away from you to ensure your elbows are directly under your shoulders when typing?	**2**	**Y**	
Do you always move your keyboard out of the way when you are using only the mouse?	**1**	**1**	
Is the keyboard clean?	**1**	**Y**	
Are the digits clear and not faded?	**1**	**Y**	
Mouse			
Is the mouse close enough to avoid extending the arm at the elbow?	**2**	**Y**	
If you have a roller ball mouse (laser mice do not need a mouse mat), do you have a mouse mat?	**1**	**N/A**	Laser Mouse
Does the mouse run freely on the mat and work accurately?	**1**	**N/A**	
Do you regularly clean your mouse and internal mouse ball?	**1**	**N/A**	
Do you reduce the time using your mouse to the lowest period possible by using keyboard short cuts?	**2**	**Y**	
Document Holder			
Do you have a document holder (if required)?	**1**	**N/A**	Not required
Can you refer to documents and papers without having to	**1**	**1**	

DSE Component		NO	Action Required / Comments
move your head?			
Other Equipment			
Is all equipment and items around you necessary? (or can it be removed to give you more desk space?)	**1**	**Y**	
Is all other equipment (phone etc) in a position to ensure that you can maintain your posture when using them?	**1**	**1**	
Space and environment			
Can you move in and out of your workstation easily?	**2**	**Y**	
Is there adequate space to manoeuvre your chair?	**2***	**Y**	
Is the area free from trailing cables which pose a trip hazard?	**2**	**Y**	
Is lighting adequate?	**2**	**Y**	
Do windows have blinds to prevent glare and reflection?	**1**	**N/A**	No windows
Do you find the work station quiet enough?	**1**	**Y**	
Is the temperature comfortable for most of the time?	**1**	**Y**	
Are you free from any upper body pain/soreness or any soreness in your hands or wrists	**2***	**Y**	
About You			
Have you had an eye test in the last 2 years? Please follow this link to eye test information	**2**	**Y**	
Do you organise your work to ensure you take a 5 minute break for every hour you are using the DSE?	**2**	**N**	
Is your workstation set up to ensure that you have a flow of work (you don't have to keep getting up or twisting for things)?	**1**	**1**	
Do you feel you understand and can effectively use all of the computer programmes you have to use as part of your job?	**2**	**N/A**	Learning different computer programmes is part of this project
Do you have an existing medical issue that you feel is being aggravated by your workstation?	**Y* = score 2**	**N**	
Do you suffer from dry or sore eyes when using your DSE?	**Y* = score 2**	**N**	
Do you feel you require extra DSE information or training?	**Y* = score 2**	**N**	
Total Score		**8**	

0 – 15 Workstation is ok, however if you have any concerns raise these with your manager

16 – 30 Contact your manager for help and advice. Are there any actions you can take easily that will improve your score (e.g. clean the screen, set up your chair?)

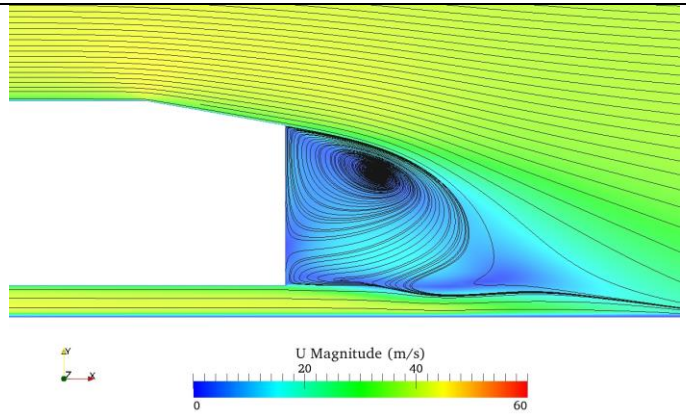
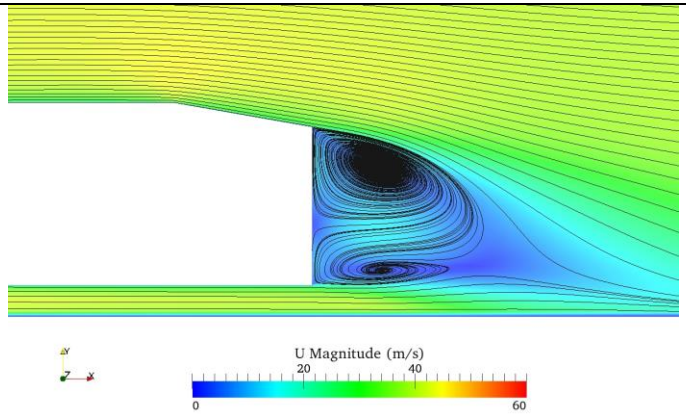
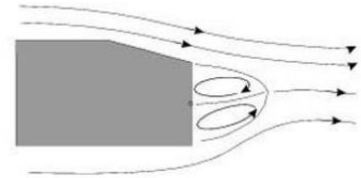
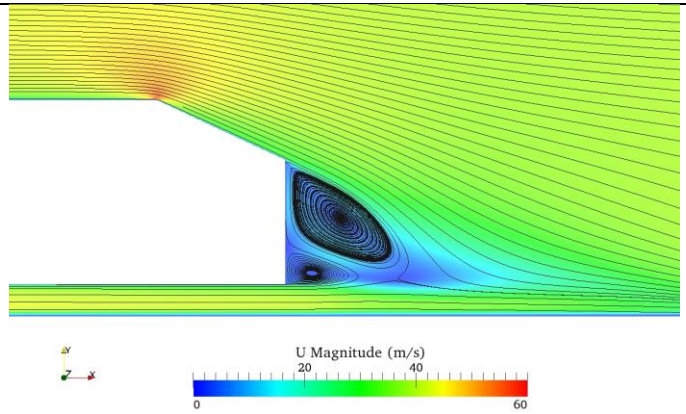
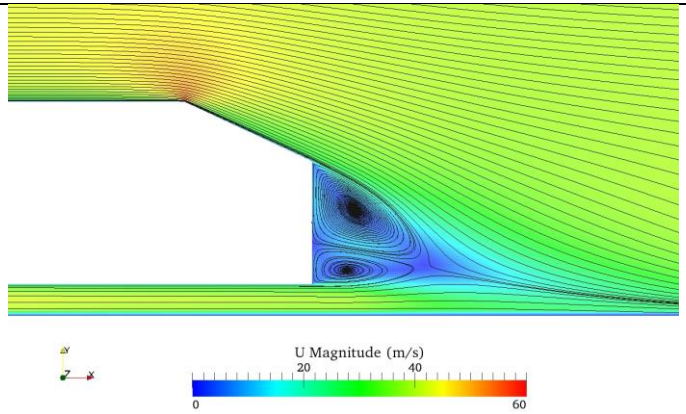
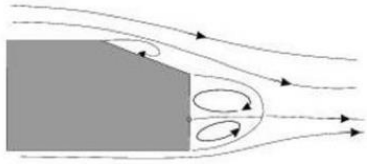
31+ Contact the **Safety Service** (safety@exeter.ac.uk) to arrange further assessment

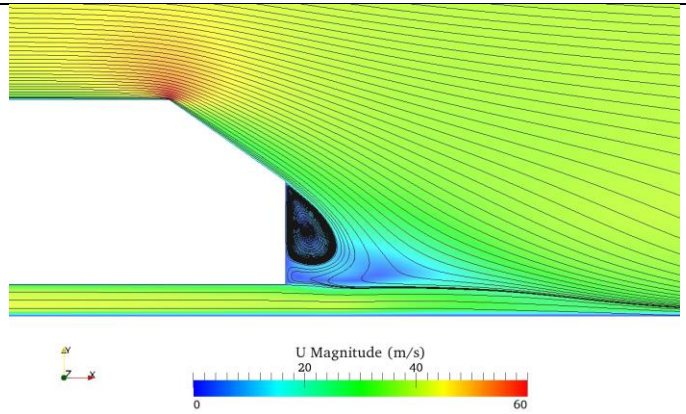
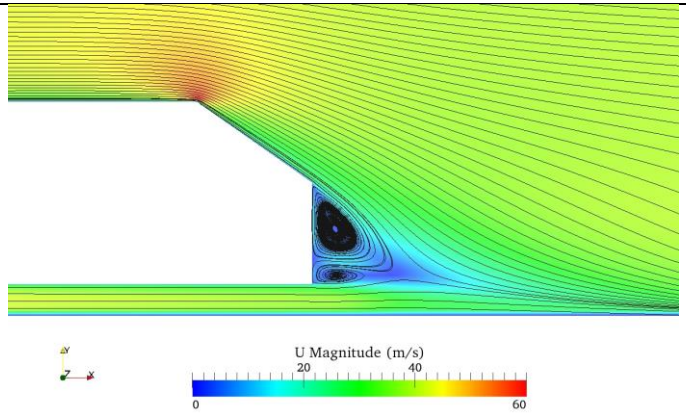
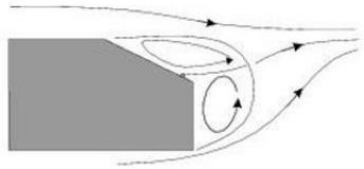
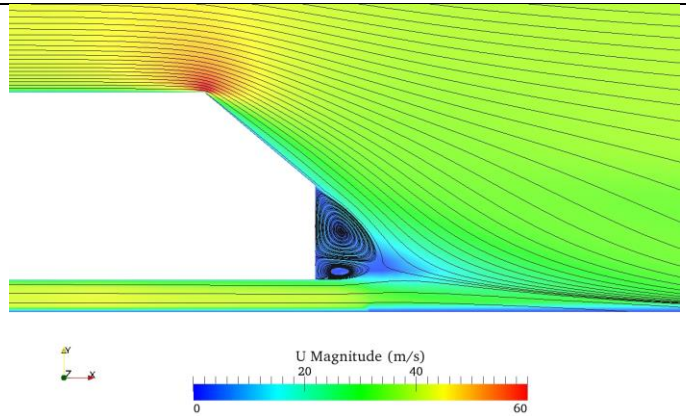
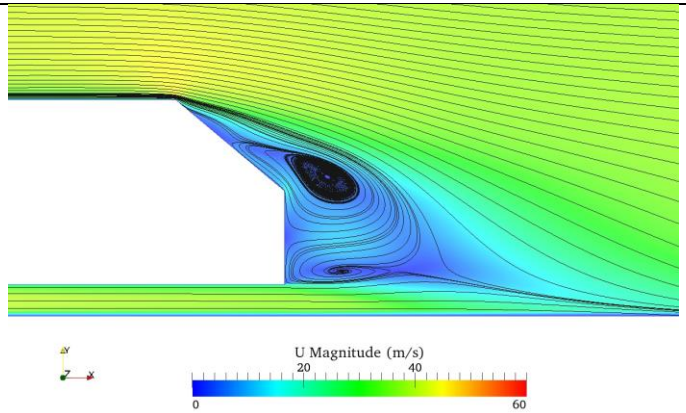
Action Plan

Please indicate in the section below what action is required to address the hazards identified in your assessment.
It is important that any key information is passed onto your line manager to ensure that action can be taken. All actions should be given a date and a person must be assigned to deal with each action. Action plans must be monitored and completed within reasonable time.

Actions Required	Responsible Person	Date for Completion
No further action required.		

Appendix B. Flow Behaviour in Wake Region

Angle (°)	k-ω SST	S-A	Wake Flow Sketch [8]
10			
25			

35	 U Magnitude (m/s) 20 40 0 60	 U Magnitude (m/s) 20 40 0 60	
40	 U Magnitude (m/s) 20 40 0 60	 U Magnitude (m/s) 20 40 0 60	

Appendix C. *Allrun* Bash Script

```
#!/bin/bash

touch running

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions #blue Room

#copy contents of 0.orig directory to 0
cp -r 0.orig 0

blockMesh > blockMesh

#surfaceFeatureExtract (for sharp edges)
surfaceFeatureExtract -includedAngle 150 constant/triSurface/AhmedBodyX.stl AhmedBodyX
>surfaceFeatureExtract

#run snappyHexMesh but overwrite the process to create final mesh files only
snappyHexMesh -overwrite > snappyHexMesh

checkMesh > checkMesh

potentialFoam >potentialFoam

cd system

rm -r forceCoeffs

mv forceCoeffs1 forceCoeffs

cd ..

simpleFoam > log.simpleFoam

foamLog log.simpleFoam > foamLog

cd logs

# plot graph of residuals
```

```
xmgrace Uy_0 Ux_0 Uz_0 p_0 nuTilda_0 -log y -legend load -world 0 0.0000001 1500 1 -hardcopy
-printfile residualsAhmedBodyX

# copy graph to Results/Residuals
cp residualsAhmedBodyX /scratch/hk282/VWT/OpenFoam/Scripting/SA/Results/Residuals

cd ..

cd forceCoeffs/0


# plot graph of drag coefficient
xmgrace forceCoeffs.dat -hardcopy -printfile forceCoeffsAhmedBodyX

# copy graph to Results/forceCoeffsGraphs
cp forceCoeffsAhmedBodyX
/scratch/hk282/VWT/OpenFoam/Scripting/SA/Results/forceCoeffsGraphs


# rename data file
mv forceCoeffs.dat forceCoeffsAhmedBodyX.dat

# copy data file to Results/forceCeoffs
cp forceCoeffsAhmedBodyX.dat /scratch/hk282/VWT/OpenFoam/Scripting/SA/Results/forceCoeffs

cd ../..

# remove running file to show it has finished
rm -f running
```

Appendix D. *FinalScript* Bash Script

```
#!/bin/bash

cd CAD

for i in `ls -1d  AhmedBody*.STL | sed -e 's.AhmedBody..' -e 's/.STL//'`

do
        mv AhmedBody$i.STL AhmedBody$i.stl #rename CAD files
done

cd ..

N=12    # Number of jobs to run in parallel

while true
do
   for f in `ls -1d CAD/AhmedBody*.stl | sed -e 's.CAD/..' -e 's/.stl//'`
   do
        if [ ! -d $f ]
        then
           running=`ls -1 AhmedBody*/running |wc -l`

           if [ $running -lt $N ]
           then
                echo 'About to run' $f
                AB=`basename $f .stl`
                cp -r template $AB

                cp CAD/$AB.stl $AB/constant/triSurface      #copy .stl file

                #rename template folder
                grep -r -l "AhmedBodyX" $AB | xargs sed -i s/AhmedBodyX/$AB/g
                cd $AB

                ./Allrun &
          sleep 1      # Sleep after starting to prevent race conditions

                cd ..
```

```
            fi
        fi
    done

    # Have all the jobs been done?
    Njobs=`ls -1d CAD/AhmedBody*.stl | wc -l`
    Nstarted=`ls -1d AhmedBody* | wc -l`
    Nrunning=`ls -1 AhmedBody*/running |wc -l`

    echo Jobs $Njobs Started $Nstarted  Running $Nrunning
    if [ $Nrunning -eq 0 -a $Nstarted -eq $Njobs ]
    then
        echo All finished
        break
    fi
    sleep 2
done

cd Results/forceCoeffs        #compile drag coefficient data for all jobs

rm -f finalForceCoeffs

for f in forceCoeffsAhmedBody*.dat
do
    echo -n -e `basename $f .dat` '\t' >> finalForceCoeffs
    tail -1 $f >> finalForceCoeffs
done
```

Appendix E. *trymlp.m* Matlab function – created by Richard Everson

```
function [Etrain, Etest, net] = trymlp(data, lambda, ntries, doplot, xtest, ttest)
% Train an MLP network with regularisation lambda
if nargin < 3, ntries = 1; end
if nargin < 4, doplot = 0;end
if nargin < 5 xtest = [] ;end

% Column 2 alpha
% Column 3 Radius

nhidden = 30;          % number of hidden units
X = data(:,2:3);       % input parameters
xbar = mean(X, 1);     % standardise data
xstd = std(X, 1);
X = (X-repmat(xbar, size(X, 1), 1))./repmat(xstd, size(X, 1), 1);
t = data(:,4);         % output parameter
Etrain = 100000;

for n = 1:ntries       % for loop to try mlp network several times
   trained = 0;
   while (trained == 0)
      try
         net = mlp(2, nhidden, 1, 'linear', lambda);     % create mlp network
         options = foptions;                             %
         %options(1) = -1;
         options(14) = 1000;
         net = netopt(net, options, X, t, 'scg');        %optimise weights in mlp network
         trained = 1;
         ytrain = mlpfwd(net, X);
         Etrainn = mean((t-ytrain).^2);                  % Squared error per data point
         if Etrainn < Etrain
            bestnet = net;
            Etrain = Etrainn;
         end
      catch
         disp('Bang')
      end
```

```
    end
end

net = bestnet;

if doplot
    Rtest = linspace(0, 50, 20);                %randomly generate 20 radii values between 0 and 50
    alphatest = linspace(5, 45, 30);            % randomly generate 30 alpha values between 5 and
45
    [aa, RR] = meshgrid(alphatest, Rtest);
    Xtest = [aa(:), RR(:)] ;
    Xtest = (Xtest - repmat(xbar, size(Xtest, 1), 1))./repmat(xstd, size(Xtest, 1), 1);      %standardise
data
    y = mlpfwd(net, Xtest);                     % use trained mlp to predict drag coefficient
    Cdtest = reshape(y, 20, 30);                % plot surface
    surf(alphatest, Rtest, Cdtest); xlabel('\alpha (degrees)'); ylabel('Radius (mm)'); zlabel ('Cd');
    v = axis;
    Cdmin = 0.24;
    Cdmax = 0.39;
    v(5) = 0.2;
    v(6) = 0.4;
    axis(v)
    set(gca, 'CLim', [Cdmin, Cdmax])
    set(gca, 'FontSize',20)
    set(findall(gcf,'type','text'),'FontSize',20,'fontWeight','bold')
    colorbar
    h=colorbar;
    set(h,'fontsize',20);
end
Etest = [];
if ~isempty(xtest)
    xtest = (xtest - repmat(xbar, size(xtest, 1), 1))./repmat(xstd, size(xtest, 1), 1);
    y = mlpfwd(net, xtest);
    Etest = mean((ttest - y).^2);
end
```

Appendix F. *loocv.m* Matlab function – created by Richard Everson

```
function [Etrain, Etest, net] = loocv(data, lambda, ntries)
%

N = size(data.data, 1);

Nlambda = length(lambda);
Etrain = zeros(Nlambda, 1);
Etest = zeros(Nlambda, 1);



for j = 1:Nlambda       % for loop to execute LOOCV
   for n = 1:N
      I = [1:n, n+1:N];   % take out row n from data set
      X = data.data(I,:);
      xtest = data.data(n, 2:3);
      ytest = data.data(n, 4);

      [Etr, Ete, net] = trymlp(X, lambda(j), ntries, 0, xtest, ytest); %run mlp function
      Etrain(j) = Etrain(j) + Etr;
      Etest(j) = Etest(j) + Ete;
   end
   fprintf(1, '%.8g  %.8g %.8g', lambda(j), Etrain(j)/N, Etest(j)/N);
end
Etrain = Etrain/N;
Etest  = Etest/N;

semilogx(lambda, Etrain, 'bo-', lambda, Etest, 'rs-')       % print Etrain and Etest against lambda

[lambda(:), Etrain(:), Etest(:)]

[emin, j] = min(Etest);             %find minimum test error
lambdabest = lambda(j);

figure;
[Etr, Ete, net] = trymlp(data.data, lambda(j), ntries, 1);  %plot mlp surface
```