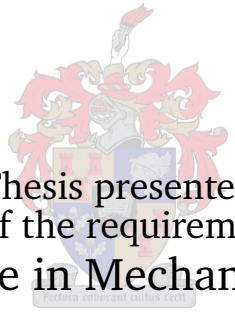
UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Optimization of a Low Speed Wind Turbine using Support Vector Regression

by

John Nathaniel Wise



Thesis presented
in partial fulfilment of the requirements for the degree of
Master of Science in Mechanical Engineering
at the

University of Stellenbosch

Department of Mechanical Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisor: Prof. G Venter

December 2008

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: .....
JN Wise

Date: .....

Abstract

NUMERICAL design optimization provides a powerful tool that assists designers in improving their products. Design optimization automatically modifies important design parameters to obtain the best product that satisfies all the design requirements. This thesis explores the use of Support Vector Regression (SVR) and demonstrates its usefulness in the numerical optimization of a low-speed wind turbine for the power coefficient, C_p. The optimization design problem is the three-dimensional optimization of a wind turbine blade by making use of four two-dimensional radial stations. The candidate airfoils at these stations are selected from the 4-digit NACA range. A metamodel of the lift and drag coefficients of the NACA 4-digit series is created with SVR by using training points evaluated with XFOIL software. These SVR approximations are used in conjunction with the Blade Element Momentum theory to calculate and optimize the C_p value for the entire blade. The high accuracy attained with the SVR metamodels makes it a viable alternative to using XFOIL directly, as it has the advantages of being faster and easier to couple with the optimizer. The technique developed allows the optimization procedure the freedom to select profiles, angles of attack and chord length from the 4-digit NACA series to find an optimal C_p value. As a result of every radial blade station consisting of a NACA 4-digit series, the same lift and drag metamodels are used for each station. This technique also makes it simple to evaluate the entire blade as one set of design variables. The thesis contains a detailed description of the design and optimization problem, the implementation of the SVR algorithm, the creation of the lift and drag metamodels with SVR and an alternative methodology, the BEM theory and a summary of the results.

Opsomming

NUMERIESE ontwerp optimering voorsien 'n kragtige hulpmiddel wat ontwerpers assisteer in die verbetering van hul produkte. Die outomatiese optimering van ontwerpe verander belangrike ontwerp parameters om die beste produk wat al die ontwerp vereistes bevredig, te verkry. Hierdie tesis ondersoek die gebruik van Ondersteunende Vektor-Passing (OVP) en bevestig die bruikbaarheid daarvan deur dit toe te pas in die numeriese optimering van 'n lae-spoed wind-turbine. Die ontwerp probleem is die drie-dimensionele optimering van 'n turbine lem met die gebruik van vier twee-dimensionele radiale stasies. Die lemprofiele vir hierdie vier stasies word gekies vanuit die 4-syfer NACA-reeks. 'n Benadering van die die hef en sleur koëffisiënte is geskep met OVP deur opleidingspunte met die sagteware XFOIL te evalueer. Hierdie OVP-benaderings is saam met die Lem Element Momentum (LEM) teorie gebruik om die drywing-koëffisiënt van die lae-spoed wind-turbine te optimeer. Die tegniek laat die optimerings prosedure toe om enige profiele van die NACA 4-syfer reeks te kies, asook om die optimale aanvalshoeke en koordlengtes te bepaal wat die C_p waarde maksimeer. Die profiel by elke radiale stasie bestaan uit 'n 4-syfer NACA lemprofiel, en die lem kan dus as 'n geheel geëvalueer word van dieselfde OVP-benaderings vir die hef en sleur koëffisiënte gebruik word. Die akkuraatheid van die benadering beteken dat dit 'n beter opsie is om te gebruik as XFOIL, omdat dit merkwaardig vinniger is en dit eenvoudiger is om aan die optimering sagteware te koppel. Hierdie tesis bevat 'n breedvoerige beskrywing van die ontwerp en optimeringprobleem, die implementasie van die OVP-algoritme, die skepping van die benaderings van die hef en sleur koëffisiënte met OVP, sowel as met 'n alternatiewe benaderings tegniek, die LEM-teorie en 'n opsomming van die uitslae.

Acknowledgment

A special word of thanks to:

- My family for support.
- Gerhard Venter for relentless encouragement and support.
- Nick for good company in the long office hours.
- Nicola Cencelli for critical assistance.
- William Hunter for his help.
- NACOE for their interest in the project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

THE numerical design optimization of engineering problems is typically computationally expensive. A method of addressing the high computational cost associated with such numerical simulations in an optimization environment is the creation of metamodels. A metamodel is an approximation of the original simulation model, effectively creating a model of the engineering model.

There are a number of advantages in using metamodels in optimization. To begin with, a metamodel can simplify the optimization process by coupling the optimizer to a function approximation, instead of coupling the optimizer to a numerical simulation or a real-life experiment. In addition, a metamodel filters out numerical noise by creating a smooth function approximation. A metamodel can also greatly reduce the time to optimize an engineering application. When an application is optimized, the optimizer typically requires a number of function calls to iteratively obtain gradient information and to perform a one dimensional search. The number of function calls may be more than the number of design points required to create a metamodel. The reduction in the required number of function calls coupled with the fact that a metamodel can also be used more than once at little additional cost can result in significant savings in computational cost.

By creating a metamodel, the computational effort is shifted from the optimization procedure to the creation of the metamodel. The metamodel obtained from a series of analyses is used to replace the computationally expensive numerical simulation during the optimization process. The time to create the metamodel is often reduced through the use of parallel computing, where many independent design points can be evaluated at the same time.

Metamodelling has been used in numerous engineering applications including aircraft

design, computational fluid dynamics and finite element analyses. In this thesis the use of Support Vector Regression (SVR) is explored as a metamodelling technique. The theory behind SVR is discussed and a number of implementation issues are dealt with in example problems. These examples demonstrate the various tasks that need to be done in order to obtain a sufficiently accurate model. A real-world application was subsequently chosen from the aerospace industry, namely the optimization of a wind-turbine, in order to explore the use of SVR in a non-linear and high dimensional analysis.

1.1 Objectives

The objective of the research represented in this thesis is twofold. The first objective is to implement SVR as a usable metamodelling technique and explore possible implementation issues. The second objective is to optimize a non-linear application using SVR as a metamodelling technology.

The non-linear application chosen for this study is the optimization of a low speed wind turbine. In this study, the objective is to maximize the power that can be extracted from the wind. This thesis draws on the work done by Cencelli [6]. The principal similarities between the current work and that of Cencelli is that the blades on the wind turbine are approximated as 4-station blades, and the Blade-Element-Momentum (BEM) method is used to find the power coefficient. Two major differences exist however. Firstly, this thesis limits the airfoil profiles used for the wind turbine to the airfoils described by the NACA 4-digit series. Secondly, in the current work, all design variables describing the turbine blade are analyzed simultaneously. In this thesis a generic method is developed in which the optimization of a wind turbine can be approached without a detailed knowledge of the lift and drag characteristics of specific airfoil types.

In this thesis the NACA 4-digit range is successfully approximated with two metamodelling techniques, Response Surface Approximations (RSA) and SVR. Once the metamodels are available, they are used to calculate the power coefficient of a low-speed wind turbine. By using the technique developed in this research, an optimizer has the liberty to choose from any of the airfoils across the NACA 4-digit range, instead of limiting it to a single profile.

1.2 Verification

In the creation of a metamodel, XFOIL [7] is used to evaluate the lift and drag coefficients for various 4-digit NACA profiles. The BEM method is then used as an intermediate analysis to find the power coefficient. The results are verified by comparing the power coefficient obtained from the BEM method with an SVR approximation to the results obtained directly with XFOIL.

Chapter 2

Optimization and Metamodelling

IN this section we introduce the concept of optimization and a look at a number of approximation techniques in order to place Support Vector Regression (SVR) in context.

2.1 Optimization

In the greater majority of regression approximations, some type of optimization is required to obtain the best fit through the data. Support vector regression is no exception, and to create an effective approximation, a number of variables need to be optimized. Optimization is also used later in the thesis to optimize the airfoil characteristics of a low-speed wind turbine. Thus the concept of optimization is introduced in this section.

A general approach to optimization is introduced as follows:

$$\begin{aligned}
\text{Minimize :} \quad & F(\mathbf{x}) && && (2.1)\\
\text{Subject to :} \quad & g_j(\mathbf{x}) \leq 0 && j = 1, m && (2.2)\\
& h_k(\mathbf{x}) = 0 && k = 1, l && (2.3)\\
& x_i^l \leq x_i \leq x_i^u && i = 1, n && (2.4)\\
\text{where} \quad & \mathbf{x} = x_1, x_2, ..., x_n && && (2.5)
\end{aligned}$$

where $\boldsymbol{x}$ represents the design variables. The approach states that the function $F(\boldsymbol{x})$ needs to be minimized subject to certain inequality constraints (Eq. 2.2) and certain equality constraints (Eq. 2.3). The side constraints in Eq. 2.4 imply upper and lower limits that are imposed on the design variables, shown in Eq. 2.5.

To find an optimum, we can use a method based on the gradient information of the objective and constraint function. These gradient-based methods are typically two step processes. Firstly, the method obtains gradient information and decides in which direction to move. A number of methods exist that calculate how to use the gradient information and previous search directions to find a new direction in which to move, for example a steepest gradient descent [1] approach. The second step is finding the optimum in the search direction, which is effectively a one-dimensional search. The search direction is defined as a search vector and is denoted as $\mathbf{S}$. Once the search vector has been found, the vector of design variables ($\mathbf{x}$) is updated according to the formula:

$$\mathbf{x}^q = \mathbf{x}^{q-1} + \alpha^* \mathbf{S}^q \tag{2.6}$$

where the product of the scalar α^* and $\mathbf{S}$ yields the optimal distance and direction from the old design point $\mathbf{x}^{q-1}$.

Gradient-based optimizers are generic optimizers that are typically used to find local optima for smooth continuous functions. This section presents the background of gradient-based optimization techniques, and discusses various techniques that are utilized in the present work.

2.1.1 Unconstrained problems

When no constraints are imposed on a problem, a minimum exists where the gradient of $F(\mathbf{x})$ vanishes as follows [1]:

$$\nabla F(\mathbf{x}) = \mathbf{0} \tag{2.7}$$

where the gradient vector $\nabla F(\mathbf{x})$ is represented as:

$$\nabla F(\mathbf{x}) = \left\{ \begin{array}{c} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_n} F(\mathbf{x}) \end{array} \right\} \tag{2.8}$$

Finding the points at which the gradient vanishes is a necessary but not sufficient condition for optimality. To check for optimality, a Hessian matrix can be constructed [1]. A Hessian matrix deals with second derivatives, and is positive definite at a local optimum. A Hessian matrix is constructed as follows [1]:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 F(\mathbf{x})}{\partial X_1^2} & \frac{\partial^2 F(\mathbf{x})}{\partial X_1 \partial X_2} & \cdots & \frac{\partial^2 F(\mathbf{x})}{\partial X_1 \partial X_n} \\ \frac{\partial^2 F(\mathbf{x})}{\partial X_2 \partial X_1} & \frac{\partial^2 F(\mathbf{x})}{\partial X_2^2} & \cdots & \frac{\partial^2 F(\mathbf{x})}{\partial X_2 \partial X_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F(\mathbf{x})}{\partial X_n \partial X_1} & \frac{\partial^2 F(\mathbf{x})}{\partial X_n \partial X_2} & \cdots & \frac{\partial F(\mathbf{x})}{\partial X_n^2} \end{bmatrix} \tag{2.9}$$

Local minima exist at the location where the gradients of $F(\mathbf{x})$ vanish and the Hessian matrix is positive definite [1]. To ensure that an optimum is the global optimum, the Hessian matrix must be evaluated at every local optimum, which is difficult and often not possible to do in practice. Starting from various initial values when using gradient-based optimization algorithms is a practical approach to help avoid local optima.

2.1.2 Constrained problems

A search vector $\mathbf{S}$ is required to search through the design space. The search vector needs to be directed in a direction that moves closer to the optimum without violating any constraints. The direction which will move closer to the optimum is defined as [1] the usable direction, while the direction which the search vector can move in without violating any constraints is defined as the feasible direction. Therefore a search vector is required that will move in both the feasible and usable direction as shown in Fig. 2.1. At the constraint boundary, the usable direction is perpendicular to the tangent of the constraint. The usable direction is mathematically represented as:

$$\mathbf{S}^T \nabla F(\mathbf{x}) \leq 0 \tag{2.10}$$

Similarly, the feasible sector can be represented as follows:

$$\mathbf{S}^T \nabla g_j(\mathbf{x}) \leq 0 \tag{2.11}$$

A number of different techniques exist for finding an efficient search direction. The research done for this thesis utilized three optimization techniques from a commercial software package called DOT [1]. These three techniques are briefly discussed in the following in this chapter.

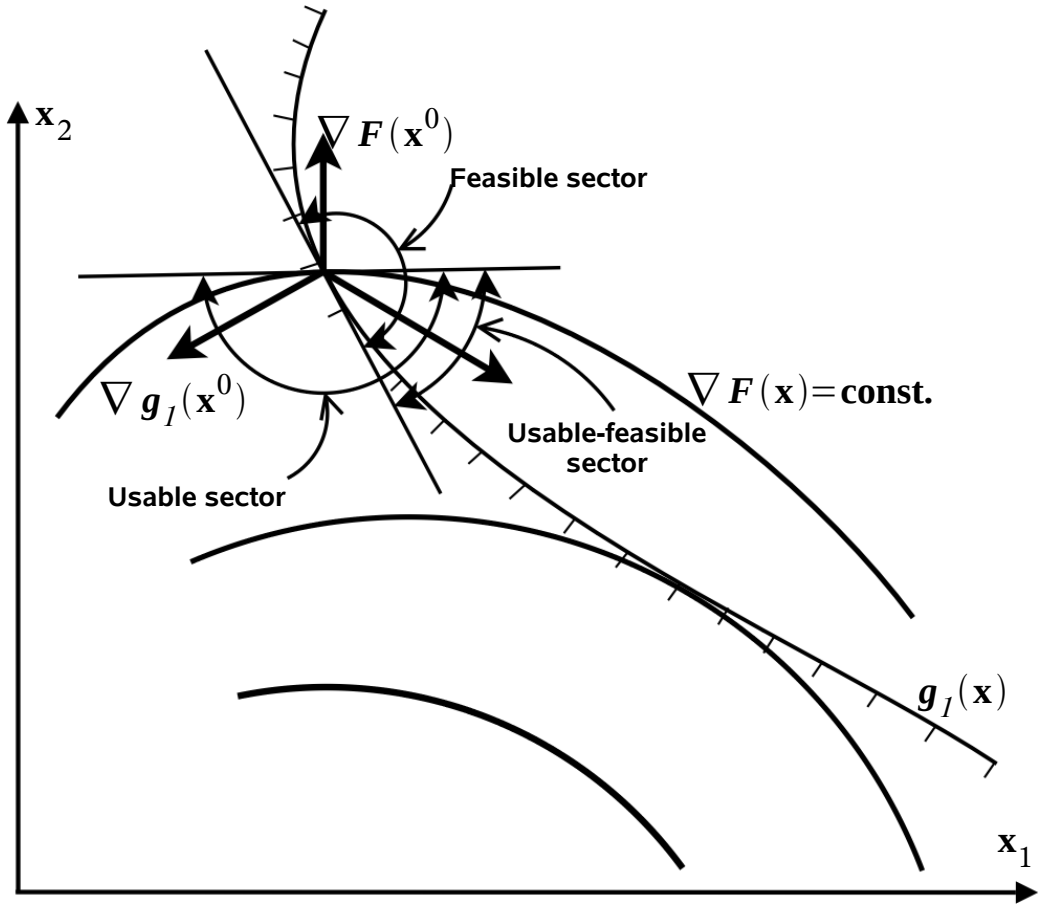


Figure 2.1: The usable and feasible search directions [1]

2.1.3 Kuhn-Tucker conditions

After searching through the design space in the search direction, an optimum is found at the point which satisfy the Kuhn-Tucker conditions [1] for an optimum. The Kuhn Tucker conditions are derived from the Lagrange [1] function:

$$L(\mathbf{x}, \lambda) = F(\mathbf{x}) + \sum_{j=1}^{m} \lambda_j g_j(\mathbf{x}) \tag{2.12}$$

where λ are variables that regulate the inequality constraints $\mathbf{g}_j$. The Kuhn-Tucker conditions provide the necessary conditions for a non-linear constrained optimal point $\mathbf{x}^*$ as follows [1]:

1. The design point $\mathbf{x}^*$ must be feasible, which means:

$$g_j(\mathbf{x}^*) \leq 0, \quad j = 1, m \tag{2.13}$$

2. The product of λ_j and $g_j(\mathbf{x}^*)$ must be equal to zero for all m inequality constraints as follows:

$$\lambda_j g_j(\mathbf{x}^*) = 0, \quad j = 1, m \tag{2.14}$$

3. The gradient of the Lagrangian must vanish at the design point:

$$\nabla F(\mathbf{x}^*) + \sum_{j=1}^{m} \lambda_j \nabla \mathbf{g}_j(\mathbf{x}^*) = 0, \quad \lambda_j \geq 0, \quad j = 1, m \tag{2.15}$$

Any point that violates a constraint will therefore not be seen as an optimum. Furthermore, any point that is evaluated for an optimum must be feasible.

2.1.4 Sequential Linear Programming (SLP)

The majority of engineering analyses are non-linear functions. However, it is often possible to find the optimum by using linear approximations around the current design point. A linear Taylor series approximation is obtained from the function value and gradient information typically obtained from finite difference gradient calculations at the current design point. The linearized optimization problem can then be stated as follows:

$$\text{Minimize :} \qquad \hat{F}(\mathbf{x}) \approx F(\mathbf{x}^0) + \nabla F(\mathbf{x^0})^T \delta \mathbf{x} \tag{2.16}$$

$$\text{Subject to :} \qquad \begin{cases} \hat{g}_j(\mathbf{x}) \approx g_j(\mathbf{x^0}) + \nabla g_j(\mathbf{x}^0)^T \delta \mathbf{x} \leq 0 & j = 1, m \\ \hat{h}_k(\mathbf{x}) \approx h_k(\mathbf{x^0}) + \nabla h_k(\mathbf{x}^0)^T \delta \mathbf{x} = 0 & k = 1, l \\ x_i^l \leq x_i^0 + \delta x_i \leq x_i^u & i = 1, n \end{cases} \tag{2.17}$$

$$\text{where} \qquad \delta \mathbf{x} = \mathbf{x} - \mathbf{x}^0 \tag{2.18}$$

The $\mathbf{x}^0$ terms defines the point about which the linearization is done. The three terms $\hat{F}$, $\hat{g}$ and $\hat{h}$ represent the linearized approximations of the objective function and constraints respectively. This optimization problem is solved by minimizing the objective function in Eq.'s 2.16, 2.17 and 2.18. The design point at which this minimum is found is evaluated and becomes the new point around which a linearization is done. This procedure is repeated iteratively until a precise solution is reached, hence the term 'sequential'.

2.1.5 Sequential Quadratic Programming (SQP)

Sequential Quadratic Programming is a technique that finds the search direction **S** by solving a quadratic sub-problem. In formulating the sub-problem for SQP the objective function is approximated by a quadratic Taylor series while the constraints are approximated by linear Taylor series.

$$\text{Minimize :} \qquad Q(\mathbf{S}) = F(\mathbf{x}) + \nabla F(\mathbf{x})^T \mathbf{S} + \frac{1}{2}\mathbf{S}^T \mathbf{B}\mathbf{S} \tag{2.19}$$

$$\text{Subject to :} \quad \begin{cases} \nabla g_j(\mathbf{x})^T \mathbf{S} + \delta_j g_j(\mathbf{x}) \leq 0 & j = 1, m \\ \nabla h_k(\mathbf{x})^T \mathbf{S} + \bar{\delta} h_k(\mathbf{x}) = 0 & k = 1, l \end{cases} \tag{2.20}$$

The symbol **B** represents a matrix that approximates the Hessian matrix [1], and the two scalar parameters $\bar{\delta}$ and δ are used to regulate and prevent inconsistencies in the linearized constraints. The **B** is updated according to a Broydon-Fletcher-Shanno-Goldfarb update formula [1]. Solving the sub-problem yields the search vector **S**, and a 1D search can commence to find the minimum. The 1D search makes use of a penalty function to search along the search vector until a minimum is reached. This procedure is implemented iteratively until the Kuhn-Tucker (Eq. 2.1.3) conditions are satisfied, or until no further improvement can be made.

2.1.6 Modified Method of Feasible Directions (MMFD)

The objective in the MMFD technique is to find a direction that reduces the objective function as rapidly as possible without violating constraints. Once a constraint is encountered, the search direction is set tangent to the constraint boundary. For a problem including only inequality constraints, the search direction is found by solving the following sub-problem [1]:

$$\text{Minimize :} \qquad \nabla F(\mathbf{x})^T \mathbf{S} \tag{2.21}$$

$$\text{Subject to :} \quad \begin{cases} \nabla g_j(\mathbf{x})^T \mathbf{S} \leq 0 & j \in J \\ \mathbf{S}^T \mathbf{S} \leq 0 \end{cases} \tag{2.22}$$

where J is the number of active constraints. An active constraint refers to constraint that is being violated. Should a constraint be violated during a 1D search, a Newton-step method is initiated to move back into the usable-feasible region. A search is followed to the point that the Kuhn-Tucker conditions are met or no further improvement can be made.

2.2 Metamodelling

A number of regression techniques exist with which metamodels can be created. Different models have different characteristics, and no single technique is the best for all regression problems. The designer that decides to use metamodelling in his or her optimization problem needs to decide how accurate the metamodel needs to be, and how much computational resources are available. A short discussion on the placement of training points, or design of experiments, is presented. Three alternative regression techniques to Support Vector Regression (SVR), namely Response Surface (RS) methodology, Radial Base Functions (RBF) and Kriging are introduced in order to put SVR in context. A basic overview of RBF's and Kriging is presented, with a more in-depth discussion on RS.

2.2.1 Design of experiments

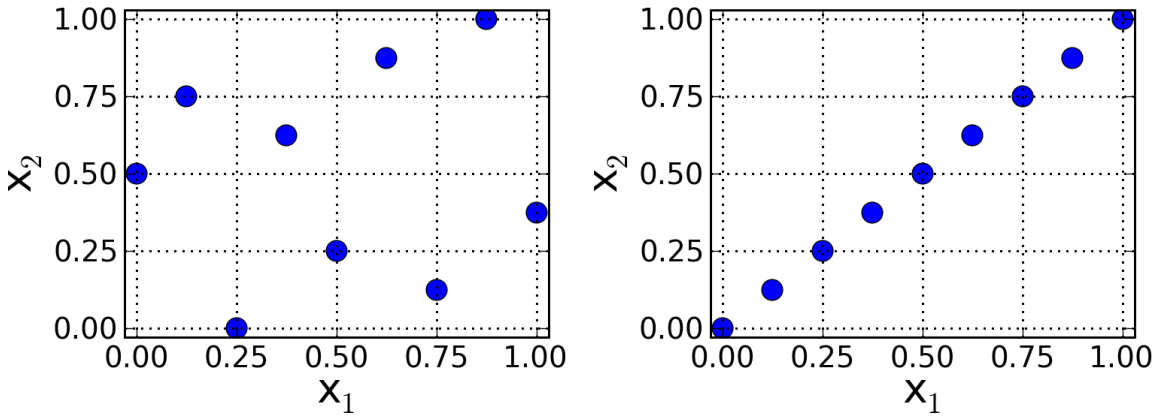
In the creation of a metamodel, a number of training points are required. The way in which these training points are created is known as a Design of Experiments (DOE). The location of these points affects the accuracy of the metamodel as well as the number of points required to create the metamodel. In the creation of a metamodel such as SVR in which no underlying model is assumed, it is generally advantageous to have a space filling DOE.

For a design of N design variables and M training points, the Latin Hypercube (LH) [8] design divides each dimension into M equal levels. For each dimension, each level is occupied by only one point. Figure 2.2a shows a typical implementation of a LH design in two dimensions. However, this design could potentially have very poor space filling characteristics, as shown in Fig. 2.2b

The Optimum Latin Hypercube (OLH) design is a better method of creating a DOE. The OLH essentially strives to maximize the minimum space between design points, while satisfying the LH requirements. In Fig. 2.3 an example of an OLH design with 9 points is presented. Here the effective space filling design properties are evident.

While the OLH has excellent space filling design properties, the creation of design points is a challenging and time consuming process. A method of approximating the OLH process has been developed by Viana, et. al. [9], called a Structured Latin Hypercube. This method approximates the effectiveness of OLH, and is done in 'minimal computational time' [9]. A variety of other DOE techniques exists. However, in this thesis only the

Optimum Latin Hypercube is utilized to create design points for metamodels and initial points.



(a) Typical implementation (b) Worst case scenario

Figure 2.2: Latin Hypercube designs

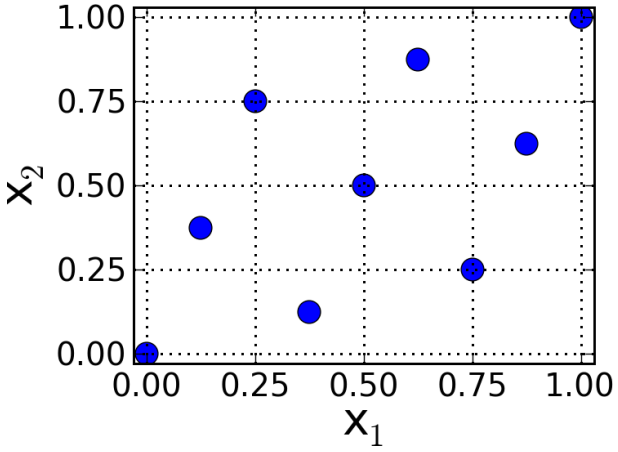


Figure 2.3: Optimum Latin Hypercube design

2.2.2 Radial basis functions (RBF's)

Radial basis approximations typically consist of a series of radial basis functions which are centred at each training point [4]. A function $y(\mathbf{x})$ can be approximated with RBF's as follows:

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{k} \lambda_i \phi(\|(\mathbf{x} - \mathbf{x_{oi}})\|) \tag{2.23}$$

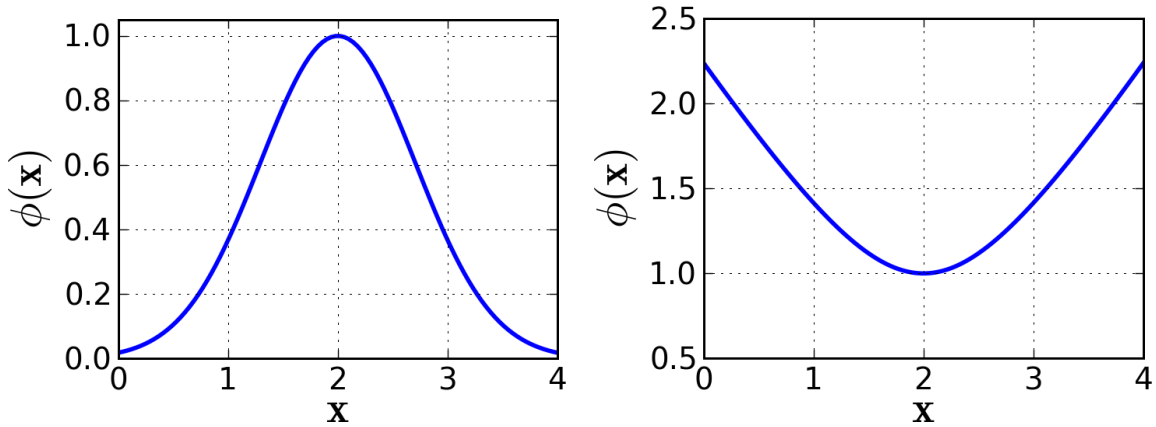
where each λ_i is an unknown weight coefficient, $\|(\mathbf{x} - \mathbf{x_{oi}})\|$ is the Euclidean norm of the distance between the design point $\mathbf{x}$ and training point $\mathbf{x_{oi}}$. The training point $\mathbf{x_{oi}}$ is known as the centre because the i^{th} RBF is centred at this training point. Furthermore, k denotes the number of training points, while the symbol ϕ denotes a user-defined radial basis function.

Table 2.1 shows two commonly used RBF's. In the RBF's a radius parameter, σ, is introduced that needs to be adjusted for an optimal approximation. The radius parameter specifies the radial extent of the basis function. The greater σ the greater the extent of radial influence of the basis function of the Gaussian function, while an increase in σ decreases the radial influence of the Multiquadratic function.

Table 2.1: Radial basis functions [4]

Gaussian	$\phi(\mathbf{x}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{x_{oi}}\|^2}{\sigma^2}\right)$
Multiquadratic	$\phi(\mathbf{x}) = \frac{\sqrt{\sigma^2 + \|\mathbf{x} - \mathbf{x_{oi}}\|^2)}}{\sigma}$

A Gaussian RBF is a monotonically decreasing function, while Multiquadratic RBF's are functions that are increasing monotonically away from the centre as shown in Fig.'s 2.4a and 2.4b respectively.



(a) Gaussian RBF function (b) Multiquadratic RBF function

Figure 2.4: Radial Basis Functions with $\mathbf{x_{oi}} = 2$ and $\sigma = 1$

The advantage of RBF approximations is that highly non-linear functions can be approximated effectively. One of the disadvantages of using RBFs to create metamodels is

that the approximation is a black box approach in which no explicit function approximation is accessible. Radial Basis Functions have been used to create approximations for non-linear systems such as aeronautic applications [4] and Finite Element Analyses [10].

2.2.3 Kriging

Kriging regression models are commonly used for engineering applications such as aerospace and mechanical engineering. These models are good options in non-linear models because they are sufficiently flexible [11] to model a broad variety of response functions. Kriging models are a combination of two functions, of which the first is a known function and the second a departure function [11]. The two functions are represented as:

$$\hat{y}(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x}) \tag{2.24}$$

where $\hat{y}(\mathbf{x})$ is the function approximating the original function, the function $f(\mathbf{x})$ is a known polynomial function and $Z(\mathbf{x})$ is the correlation function. The polynomial function is often taken as a constant. Kriging models are also well suited to modelling non-linear models. The disadvantage of Kriging is that it is also a black box approach in which no explicit function approximation is directly accessible.

2.2.4 Response surface methodology

Response Surface Approximations (RSA) is a methodology [12] that approximates functions by fitting a low order polynomial over a given number of design points. This polynomial approximation assumes an underlying trend in the data. A polynomial approximation is created by using a least squares fit over a number of training points. Some emphasis is placed on RS approximations as this methodology is implemented and used to compare the results obtained from the SVR metamodels.

Linear approximation

A first order polynomial is used for approximating linear or low curvature functions [12]. Linear approximations are represented as:

$$\hat{y}(\mathbf{x}) = b_o + \sum_{i=1}^{n} b_i x_i \tag{2.25}$$

where b_o represents the offset and all b_i terms represent gradients in each dimension. The n value represents the number of design variables.

Second order approximation

Second order polynomials include linear, quadratic and all two-factor terms. These polynomial approximations are used to model data that is assumed to follow an underlying 2^{nd} order trend. The advantage of a 2^{nd} order polynomial fit is that it only yields one optimum, which is a global optimum. The general equation describing a 2^{nd} order RS approximation is described as follows [12]:

$$\hat{y}(\mathbf{x}) = b_o + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} b_{ii} x_i^2 + \sum_{1 \le i < j \le n} \sum b_{ij} x_i x_j \tag{2.26}$$

Estimating the parameters

The parameters that need to be estimated are the **b** values. This section presents a least-squares approach [12] to solving a Response Surface problem. For the sake of clarity, a linear model is discussed here as shown in Eq. 2.25. However, the theory follows a similar approach for any z^{th} order approximation, where $z \in \mathbf{R}$. The training points can be described as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{2.27}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{k1} & x_{k2} & \cdots & x_{kn} \end{bmatrix},$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} \text{ and } \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_k \end{bmatrix}$$

Here the **x** and **y** are the training points, $\boldsymbol{\beta}$ is vector of unknown regression coefficients and $\boldsymbol{\varepsilon}$ is a vector of the errors at each training point. A vector, **b**, is required that approximates the unknown $\boldsymbol{\beta}$ values. The least-squares approach requires that the estimator **b**

is found that minimizes the sum of the square of the errors as follows:

$$L = \sum_{i=1}^{n} \varepsilon_i^2 = \boldsymbol{\varepsilon}'\boldsymbol{\varepsilon} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \tag{2.28}$$

When the terms are multiplied out, L is expressed as:

$$L = \mathbf{y}'\mathbf{y} - \boldsymbol{\beta}'\mathbf{X}'\mathbf{y} - \mathbf{y}'\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}'\mathbf{X}'\mathbf{X}\boldsymbol{\beta} \tag{2.29}$$

$$= \mathbf{y}'\mathbf{y} - 2\boldsymbol{\beta}'\mathbf{X}'\mathbf{y} + \boldsymbol{\beta}'\mathbf{X}'\mathbf{X}\boldsymbol{\beta} \tag{2.30}$$

The simplification in the second step is valid since the scalar $\boldsymbol{\beta}'\mathbf{X}'\mathbf{y}$ is equal to its transpose $(\boldsymbol{\beta}'\mathbf{X}'\mathbf{y})' = \mathbf{y}'\mathbf{X}\boldsymbol{\beta}$. A quadratic problem emerges that has its optimum at some point described by the estimator $\mathbf{b}$. The minimum of this quadratic problem can be found by setting the derivative equal to zero as follows:

$$\left.\frac{\partial L}{\partial \boldsymbol{\beta}}\right|_{\mathbf{b}} = -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\mathbf{b} = 0 \tag{2.31}$$

which is simplified to

$$\mathbf{X}'\mathbf{X}\mathbf{b} = \mathbf{X}'\mathbf{y} \tag{2.32}$$

To find the least-squares estimator of $\boldsymbol{\beta}$, both sides of Eq. 2.32 are multiplied by the inverse $\mathbf{X}'\mathbf{X}$.

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{2.33}$$

The regression model can finally be expressed as

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{b} \tag{2.34}$$

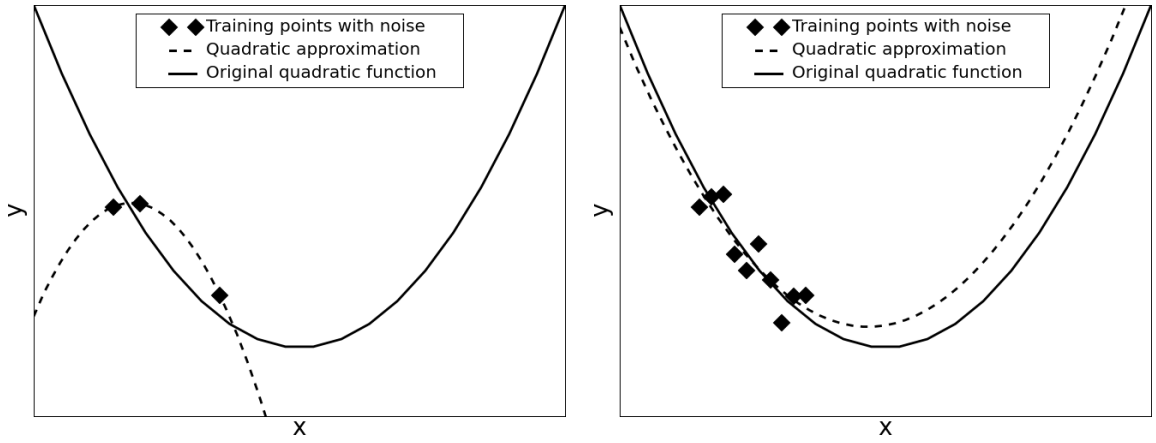
The linear and 2^{nd} order models shown in Eq.'s 2.25 and 2.26 are derived from Eq. 2.34. With this theory it is possible to create any polynomial approximations from a set of training points.

Number of design points

The minimum number of points, k_{min}, required to solve all the coefficients in a second order polynomial are calculated as follows:

$$k_{min} = \frac{(n+1)(n+2)}{2} \tag{2.35}$$

where n is the number of design variables or dimensionality of the problem. From Eq. 2.35 we can determine that a function of 5 variables for example will require at least 21 points. Using the minimum number of points means that all noise in the training points will be taken into account. An exaggerated example of the effect of noise is shown in Fig. 2.5a. To reduce the effect of noise, a multiple of k_{min} is typically used to create the approximation. In Fig . 2.5b, 12 training points ($= 4 \times k_{min}$) were used, which results in a more realistic approximation of the original function.



(a) Using $k_{min} = 3$ training points (b) Using $4 \times k_{min} = 12$ training points

Figure 2.5: Quadratic approximation

Furthermore, when a polynomial fit is used the designer assumes that the underlying function describing the data follows a second order trend. If this assumption is incorrect, the original data cannot be well represented. The advantage of using a second order fit is that only a single, and thus also a global, optimum always exists.

Chapter 3

Support Vector Regression

FOLLOWING on from some well-known metamodelling techniques presented in Chapter 2, this chapter presents the theory and examples explaining the relatively new metamodelling technology, namely Support Vector Regression.

3.1 Developing the optimization problem

Support Vector Regression (SVR) finds its niche in approximating models that are highly non-linear. One of the advantages of a SVR approximation is that it does not assume an underlying model. Another advantage of SVR is that is does not follow the 'black box' approach for Kriging and RBF approximations. In this section some of the fundamental developments of SVR are presented [2].

3.1.1 Linear Support Vector Regression

We begin by describing the case of a linear function. The initial objective is to find a function that approximates the original function from which a number of training points $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$ are taken. The approximation must have at most a ε deviation from the training points, while having a gradient that is as flat as possible. The function can be described as follows:

$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b \tag{3.1}$$

Here $\langle \mathbf{w} \cdot \mathbf{x} \rangle$ is a reference to the dot product between $\mathbf{w}$ and $\mathbf{x}$. In the linear model, flatness means that a small $\mathbf{w}$ is sought. An optimization problem is then created in

which both the flatness and the ε deviation is addressed.

$$\text{Minimize :} \quad \frac{1}{2}|\mathbf{w}|^2 \tag{3.2}$$

$$\text{Subject to :} \quad \begin{cases} y_i - \langle \mathbf{w} \cdot \mathbf{x_i} \rangle - b \leqslant \varepsilon \\ \langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b - y_i \leqslant \varepsilon \end{cases} \tag{3.3}$$

In this formulation it is assumed that there is a function described by $\mathbf{w}$ that can fit a regression over all training data within a ε error. However, this is quite an inflexible approach because we want to allow for noise or 'outlying' data points. Slack variables, ξ and ξ^*, are thus introduced and the optimization problem as stated in Smola [2] is now formulated as:

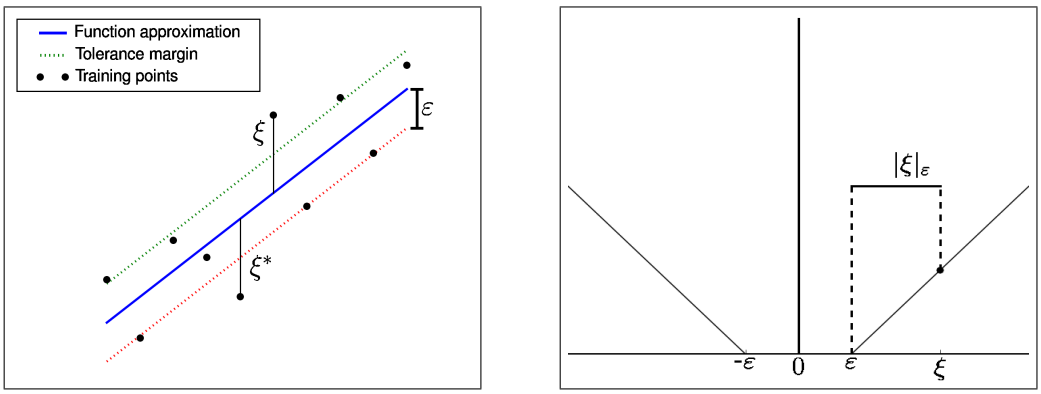
$$\text{Minimize :} \quad \frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{k}(\xi_i + \xi_i^*) \tag{3.4}$$

$$\text{Subject to :} \quad \begin{cases} y_i - \langle \mathbf{w} \cdot \mathbf{x_i} \rangle - b \leqslant \varepsilon + \xi_i \\ \langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b - y_i \leqslant \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{3.5}$$

The variable C determines the trade-off between flatness and the degree to which errors larger than ε are tolerated. By increasing C, the ξ and ξ^* errors are effectively 'amplified', causing the optimizer to avoid them. The design variables ω, b, ξ_i and ξ_i^* in the optimization problem are known as the *primal* variables, and a dual formulation introducing dual variables will be introduced in the following subsection.

In Fig. 3.1a the two dashed lines are each at a distance of ε from the optimal regression fit. The margin that allows for outlying data is known as a soft margin, and is shown in Fig. 3.1b. A number of techniques exist for implementing a soft margin, and the one that is presented here is known as ε-insensitive SVR. This soft margin penalty can be represented mathematically as:

$$|\xi|_\varepsilon = \begin{cases} 0 & \text{if } |\xi| \leqslant \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases} \tag{3.6}$$



(a) Regression with outliers (b) Soft margin penalty

Figure 3.1: Implementing the soft margin [2]

3.1.2 Dual formulation

A Lagrangian function [13] can be constructed by combining the optimization problem of Eq. 3.4 and the inequality constraints of Eq. 3.5. It is a particularly useful formulation because the optimum of the Lagrangian also yields the optimum of the original design problem. The Lagrangian is a combination of the objective function, the inequality constraints 2.2 and the equality constraints 2.3. The Lagrangian is represented as follows:

$$L(\mathbf{x}, \lambda) = F(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{l} \lambda_{m+j} h_k(\mathbf{x}) \tag{3.7}$$

The λ's are introduced as the Lagrange multipliers. The variables m and k represent the number of inequality and equality constraints respectively. A more detailed description of the Lagrangian function can be found in Vanderplaats [1]. The Lagrangian for the Linear Support Vector objective function and constraints are represented as follows:

$$\begin{aligned} L =& \frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{k}(\xi_i + \xi_i^*) - \sum_{i=1}^{k}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\ & - \sum_{i=1}^{k}\alpha_i(\varepsilon + \xi_i - y_i + \langle \mathbf{w}.\mathbf{x_i}\rangle + b) \\ & - \sum_{i=1}^{k}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle \mathbf{w}.\mathbf{x_i}\rangle - b) \end{aligned} \tag{3.8}$$

Here L is the Lagrangian function, η_i, η_i^*, α_i and α_i^* are the Lagrange multipliers, or *dual* variables, and l is the number of training samples. The Lagrangian is constructed from the objective function, alternately known as the *primal* objective function. A saddle point exists with respect to the primal and dual variables at the solution. Quoting from

Schölkopf [14]:

> 'The Lagrangian has to be minimized with respect to the primal variables **w** and b and maximized with respect to the dual variables α.'

The existence of a saddle point means that a global optimum can always be found. At the saddle point the partial derivatives with respect the primal variables ($\mathbf{w}$, b, ξ_i, ξ_i^*) have to vanish for optimality [14]. At the saddle point, the following equations hold true:

$$\partial_b L = \sum_{i=1}^{k} (\alpha_i^* - \alpha_i) = 0 \tag{3.9}$$

$$\partial_w L = w - \sum_{i=1}^{k} (\alpha_i - \alpha_i^*) x_i = 0 \tag{3.10}$$

$$\partial_{\xi_i^{(*)}} = C - \alpha_i^{(*)} - \eta_i^{(*)} \tag{3.11}$$

Substituting 3.9, 3.10 and 3.11 into 3.8 finally yields the dual optimization problem:

$$\text{Minimize :} \quad \begin{cases} -\frac{1}{2} \sum_{i,\, j=1}^{k} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(\mathbf{x_i}, \mathbf{x_j}) \\ -\varepsilon \sum_{i=1}^{k} (\alpha_i - \alpha_i^*) + \sum_{i=1}^{k} y_i (\alpha_i - \alpha_i^*) \end{cases} \tag{3.12}$$

$$\text{Subject to :} \quad \begin{cases} \sum_{i=1}^{k} (\alpha_i - \alpha_i^*) \\ (\alpha_i, \alpha_i^*) \in [0, C] \end{cases} \tag{3.13}$$

In deriving this dual formulation, the dual variables η_i and η_i^* are eliminated and are not dealt with any further. However, the **w** or weight vector derived from Eq. 3.10 is now rewritten with the remaining dual variables as follows:

$$\mathbf{w} = \sum_{i=1}^{k} (\alpha_i - \alpha_i^*) \mathbf{x}_i \tag{3.14}$$

This shows that the weight vector becomes a function of the training inputs, hence the name 'Support Vectors'. The weight vector is substituted back into the original linear SVR approximation, Eq. 3.1. The SVR function approximation is now represented as:

$$f(\mathbf{x}) = \sum_{i=1}^{k} (\alpha_i - \alpha_i^*) \langle \mathbf{x}_i, \mathbf{x} \rangle + b \tag{3.15}$$

Here it can be seen that the function approximation at any design point $\mathbf{x}$ is a function of the training points $\mathbf{x}_i$, and the weight vectors α_i and α_i^*.

3.1.3 Kernels

To make the Support-Vector algorithm non-linear, we map the data to a different feature space. Here we introduce a technique in which data is mapped and approximated in a different feature space. The change in the representation of the data can be written mathematically as:

$$\mathbf{x} = (x_1, \ldots, x_n) \longmapsto \boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_N(\mathbf{x})) \tag{3.16}$$

Equation 3.16 shows the data being mapped from the *input space* X into a *feature space,* F, where

$$F = \boldsymbol{\phi}(\mathbf{x}) \mid \mathbf{x} \in X \tag{3.17}$$

Mapping to a different feature space means that the optimization problem becomes finding the flattest function in the feature space, and no longer in the input space. The principal advantage of mapping is that it reduces the dimensionality of the problem. This characteristic was initially developed for Support Vector Classification [15] where the objective is to separate or classify training data with different attributes.

In Table 3.1 some of the different kernels that can be used in an SVR approximation are introduced. The linear kernel used in Eq. 3.15 is listed first, and is used for linear regression. The other kernels are used for linear and non-linear approximations alike. Each of the non-linear kernels has a number of parameters that need to be optimized. To guarantee a unique optimal solution, the kernel matrix $\mathbf{K}_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$ that is utilized must be positive definite [15].

Table 3.1: Kernel types [5]

Linear	$k(\mathbf{x_1}, \mathbf{x_2}) = \langle \mathbf{x_1} \cdot \mathbf{x_2} \rangle$
Polynomial	$k(\mathbf{x_1}, \mathbf{x_2}) = \langle \mathbf{x_1} \cdot \mathbf{x_2} \rangle^d$
Gaussian	$k(\mathbf{x_1}, \mathbf{x_2}) = e^{\left(\frac{-\lVert \mathbf{x_1} - \mathbf{x_2} \rVert^2}{2\sigma^2} \right)}$
Sigmoid	$k(\mathbf{x_1}, \mathbf{x_2}) = \tanh(\kappa \langle \mathbf{x_1} \cdot \mathbf{x_2} \rangle + \vartheta)$
Inhomogeneous polynomial	$k(\mathbf{x_1}, \mathbf{x_2}) = (\langle \mathbf{x_1} \cdot \mathbf{x_2} \rangle + c)^d$

A Gaussian Kernel shown in Table 3.1, also known as a Gaussian Radial Basis Function

(GRBF) [14] is utilized in this project. Gaussian Radial Basis Kernels are commonly used for creating SVR approximations through data [14], [16]. The data that is approximated with SVR using a GRBF is assumed to be a smooth function. When the kernel function is applied to Eq. 3.15, an SVR approximation can finally be represented as:

$$f(\mathbf{x}) = \sum_{i=1}^{k} (\alpha_i - \alpha_i^*) k(\mathbf{x_i}, \mathbf{x_j}) + b \tag{3.18}$$

It is interesting to note that by using a kernel function, a non-linear approximation is still dealt with in a linear space [14].

3.1.4 Final SVR optimization problem

The kernel function can now be substituted into the equation describing the approximation function as well as the optimization problem as follows:

$$\text{Minimize :} \quad \begin{cases} -\frac{1}{2} \sum_{i,\, j=1}^{k} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(\mathbf{x_i}, \mathbf{x_j}) \\ -\varepsilon \sum_{i=1}^{k} (\alpha_i - \alpha_i^*) + \sum_{i=1}^{k} y_i (\alpha_i - \alpha_i^*) \end{cases} \tag{3.19}$$

$$\text{Subject to :} \quad \begin{cases} \sum_{i=1}^{k} (\alpha_i - \alpha_i^*) \\ (\alpha_i, \alpha_i^*) \in [0, C] \end{cases} \tag{3.20}$$

In this formulation the **w** vector is no longer explicitly given as in the primal case. It is now possible to approximate data with either linear or non-linear characteristics.

3.1.5 Outline

The steps that typically need to be followed in creating an approximation with SVR are shown in Figure 3.2. This flowchart assumes a priori specification of the kernel and its parameters, and the optimization parameters C and ε. The figure shows a number of training points that are given as input. The dual variables are found with some optimization technique. Once the dual variables have been found, the offset can be determined and the final approximation for the original function can be created.

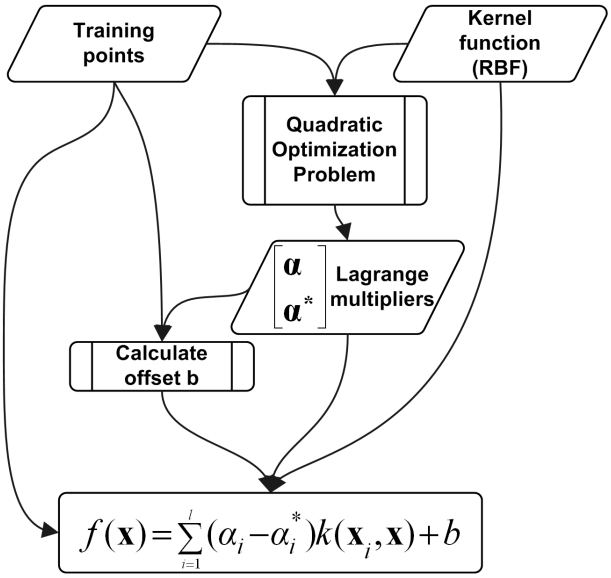


Figure 3.2: Flowchart of a typical SVR implementation

3.2 Further development of SVR

The essential components of SVR have been discussed, and in this section further implementation aspects are presented.

3.2.1 Computing the offset

The bias, or offset value, can be calculated by exploiting the Karush-Kuhn-Tucker [2] conditions. These conditions state that the product between the Lagrange multipliers (or dual variables) and the constraints must vanish to zero as follows:

$$\alpha_i(\varepsilon + \xi_i - y_i + \langle \mathbf{w}, \mathbf{x_i} \rangle + b) = 0 \tag{3.21}$$

$$\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle \mathbf{w}, \mathbf{x_i} \rangle - b) = 0 \tag{3.22}$$

and

$$(C - \alpha_i)\xi_i = 0 \tag{3.23}$$

$$(C - \alpha_i^*)\xi_i^* = 0 \tag{3.24}$$

It is apparent that either the Lagrange multipliers or the bracketed terms must vanish. Here two characteristics [2] of the optimization problem are highlighted. Firstly, wherever ξ_i or ξ_i^* is not zero, the corresponding α_i or α_i^* is equal to C. This means that

each sample $(\mathbf{x}_i, y_i)$ that has a $\alpha_i = C$ or $\alpha_i^* = C$ lies outside the ε-insensitive region. Secondly, the product of the dual variables is zero, $\alpha_i \alpha_i^* = 0$. Also, there are no sets of dual variables α_i, α_i^* that are both simultaneously non-zero. As a result of these two characteristics, the following [2] can be derived for the offset b:

$$\varepsilon - y_i + \langle \mathbf{w}, \mathbf{x_i} \rangle + b \geq 0 \quad \text{and} \quad \xi_i = 0 \qquad \text{if} \quad \alpha_i < C \tag{3.25}$$

$$\varepsilon - y_i + \langle \mathbf{w}, \mathbf{x_i} \rangle + b \leq 0 \qquad \text{if} \quad \alpha_i > 0 \tag{3.26}$$

If α_i^* is included in the analogy [2], a constraint is imposed on b as follows:

$$\begin{aligned} \max\{-\varepsilon + y_i - \langle \mathbf{w}, \mathbf{x} \rangle | \alpha_i < C \text{ or } \alpha_i^* > 0\} \leq b \leq \\ \min\{-\varepsilon + y_i - \langle \mathbf{w}, \mathbf{x} \rangle | \alpha_i > 0 \text{ or } \alpha_i^* < C\} \end{aligned} \tag{3.27}$$

which finally allows us to determine the range of bias values. In the interior point algorithm [13] utilized for this thesis, the bias is a by-product of the optimization process.

3.2.2 Quadratic problem solver

Quadratic Problem (QP) solvers are derived specifically for the optimization of quadratic problems. The optimization problem for SVR has been shown to be a QP. A QP optimizer written by Vanderbei [13], called LOQO, was used for creating the SVR approximations. The LOQO optimizer solves quadratic problems with an Interior Point algorithm. The problem to be solved is as follows [17]:

$$\text{Minimize :} \quad \frac{1}{2}\mathbf{z}^T \mathbf{H} \mathbf{z} + \mathbf{c}^T \mathbf{z} \tag{3.28}$$

by varying the value of $\mathbf{z}$, where

$$\mathbf{H} = \begin{bmatrix} \mathbf{X}\mathbf{X}^T & -\mathbf{X}\mathbf{X}^T \\ -\mathbf{X}\mathbf{X}^T & \mathbf{X}\mathbf{X}^T \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \varepsilon + \mathbf{Y} \\ \varepsilon - \mathbf{Y} \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{bmatrix} \tag{3.29}$$

The symbols $\mathbf{X}$ and $\mathbf{Y}$ contain the training points and are defined as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_l \end{bmatrix}, \qquad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_l \end{bmatrix} \tag{3.30}$$

Furthermore, the constraints on z and the α values are

$$z.\,[1,\ldots,1,-1,\ldots,-1] = 0, \qquad \alpha_i,\ \alpha_i^* \geq 0, \quad i = 0,1,\ldots,l \tag{3.31}$$

These matrices are processed by the QP optimizer. The optimizer returns the α's and bias values which describe the optimal regression approximation for a set of training points and a specified kernel parameters.

3.2.3 Parameter optimization

All kernels, with the exception of the linear kernel, have parameters that need to be adjusted to minimize the error between the SVR approximation and the data. The Gaussian Radial Base Function (GRBF) kernel has two parameters, namely the C and the σ values. The techniques that have been developed to find these parameters have been developed primarily for use in Support Vector Classification [15]. The most common technique of finding the optimal parameters for a chosen kernel is a grid search [16]. This entails an iterative grid search until the smallest error between the function evaluation and the experimental data is found. The search starts at small values of C and σ, typically 2^{-5+2k} and 2^{-15+2k} respectively, where k is equal to one and increases until suitable values are found. A more refined search can then be done in the area of the suitable parameters.

To evaluate the accuracy of the regression approximation with chosen kernel parameters, a separate set of data is needed in addition to the data used for training the regression approximation. The separate data is evaluated with the original function and with the metamodel, and so determines the accuracy of the metamodel at points away from the training points. One method of verifying the accuracy of the metamodel is to create a separate data set used solely for this purpose. However, this option is not always feasible because of computational or monetary costs involved in simulations. One alternative method is a cross validation technique. In υ-fold cross validation [16], the data is split into a υ groups. One υ group is used as 'experimental' data and the remaining $\upsilon - 1$ groups are used to train the regression approximation. Each group is iteratively taken as the experimental data and the best average parameters are chosen. In this work a separate data set is used to determine the error.

One other parameter in the SVR optimization problem that needs to be set is ε, which is set to 10^{-4} for all problems unless otherwise specified. By having the value of $\varepsilon = 10^{-4}$ means that an error of up to 0.1% is ignored on the training points, according to the

ε-insensitive SVR algorithm..

To determine the accuracy of some function approximation, $\hat{f}(\mathbf{x}_i)$ and the original function that is being approximated, $f(\mathbf{x})$, a form of error evaluation is needed. Three different methods in which to determine errors on an approximation are shown in Table 3.2. In the equations n_{errors} signifies the number of experimental data points and $\max \|\hat{f}(\mathbf{x}) - f(\mathbf{x})\|$ is the absolute difference between the original and approximated function.

Table 3.2: Error equations [5]

Maximum absolute error (MAE)	$\max \lvert\hat{f}(\mathbf{x})_i - f(\mathbf{x})_i\rvert, \quad \mathrm{i} = 1, 2, \ldots, n_{errors}$
Average absolute error (AAE)	$\frac{1}{n_{errors}} \sum_{i=1}^{n} \lvert\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i)\rvert$
Root mean square error (RMSE)	$\sqrt{\frac{\sum_{i=1}^{n} (\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i))^2}{n_{errors}}}$

3.2.4 Scaling the data

When dealing with a function of more than one parameter, a new problem can potentially occur - namely the scale difference of the dimensions. When design variables have orders of magnitude difference in their ranges, the accuracy can be adversely affected due to the radial differences that will be required when using a GRBF kernel. Scaling is done to improve the numerical accuracy of the optimization process.

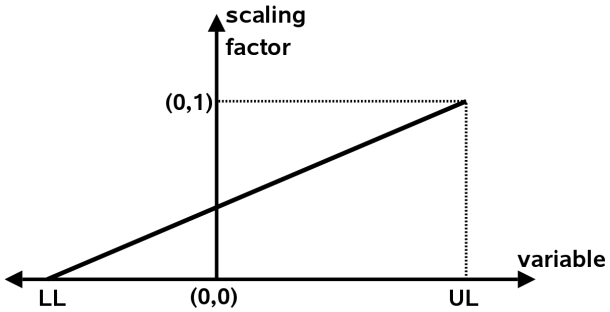


Figure 3.3: Scaling variables from upper and lower boundaries to 0 and 1

An example of scaled data is shown in Fig. 3.3. An array of data between upper and lower limits (UL and LL) is scaled to $[0, 1]$ before creating a regression approximation. Scaling the data allows the regression to approximate data of the same order of magnitude in each dimension. Scaling can therefore result in better approximation of the data.

3.3 Example implementations

This section explores two regression problems in order to ascertain the problems that need to be overcome when creating a SVR approximation. The problem of finding optimal parameters for a Gaussian Radial Basis Function (GRBF) is explored for functions of one and two variables. The following sections explore the effect of the σ (in the GRBF kernel) and the C (used in the general SVR optimization problem Eq.3.12) in example problems. In these examples the effect of the various design parameters are explored.

3.3.1 Approximating a function of one variable

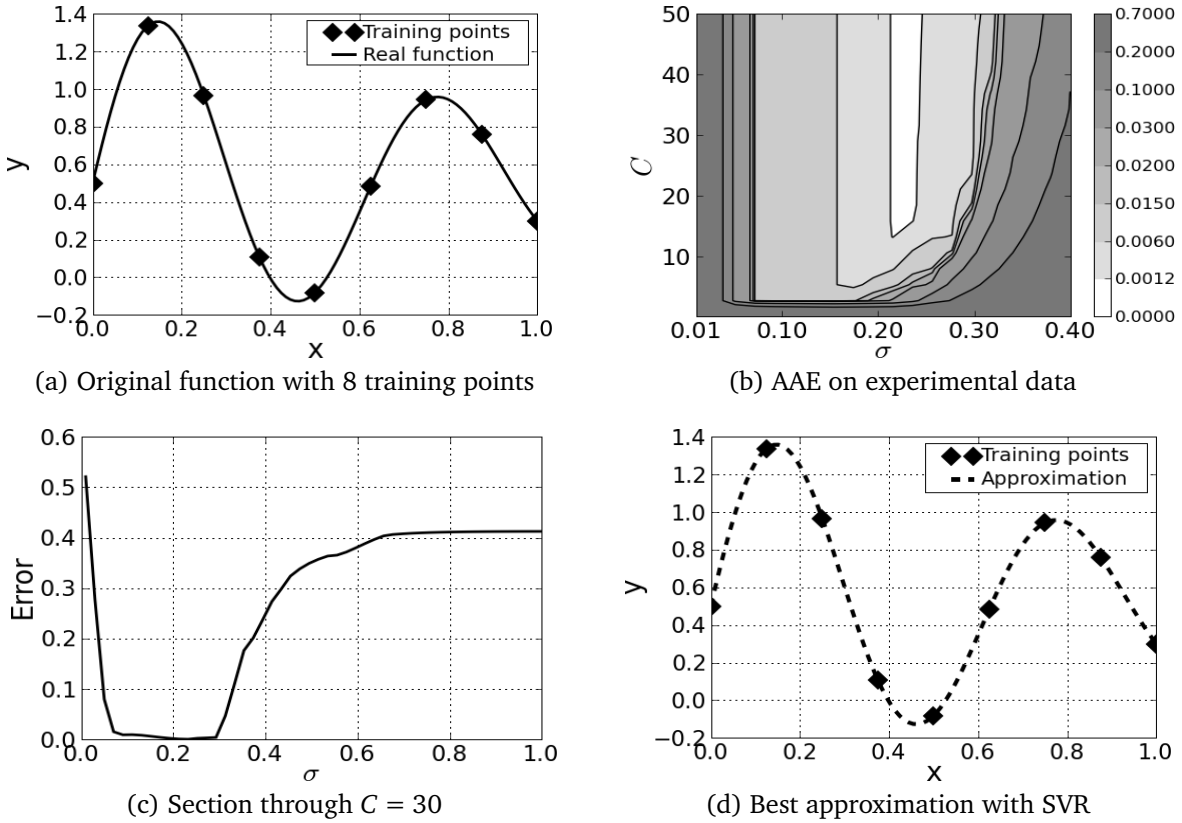
To determine the effects of varying the parameters of the GRBF kernel, a function of one variable is analyzed. The function can be described by Eq. 3.32 below.

$$f(x) = \sin(10x)e^{-x} + 0.5 \tag{3.32}$$

Eight training points are extracted from this function, shown in Fig. 3.4, and approximated with SVR. Another 40 points of the original function are evaluated function, and are used as experimental data to verify the accuracy of the regression approximation. The objective is to minimize the error between the regression fit and the 40 experimental points. The best approximation found is shown in Fig. 3.4d. The AAE (from Table 3.2) between the SVR approximation and the experimental data is evaluated by varying the parameters σ and C, as shown in Fig. 3.4b. It can be seen that an area exists in which the error between the experimental data and the function approximation is minimized.

It is also apparent from Fig. 3.4 that both parameters C and σ require adjustment to optimize the approximation. Here the value of ε is set to 10^{-4}, which results in a 'valley' in the C and σ contour plot as shown in Fig. 3.4c. In the contour plot a minimum error band exists that doesn't vary significantly when a sufficiently high value of C is

used. This means that when we exceed a certain value of C an improvement in the SVR approximation cannot be made. However, it is necessary to find the minimum point in the valley by varying the σ value. To analyze this problem in greater detail, a section is taken through the contours at $C = 30$ as seen in Fig. 3.4c. This shows the band from which a σ value must be selected to create an accurate approximation.



(a) Original function with 8 training points

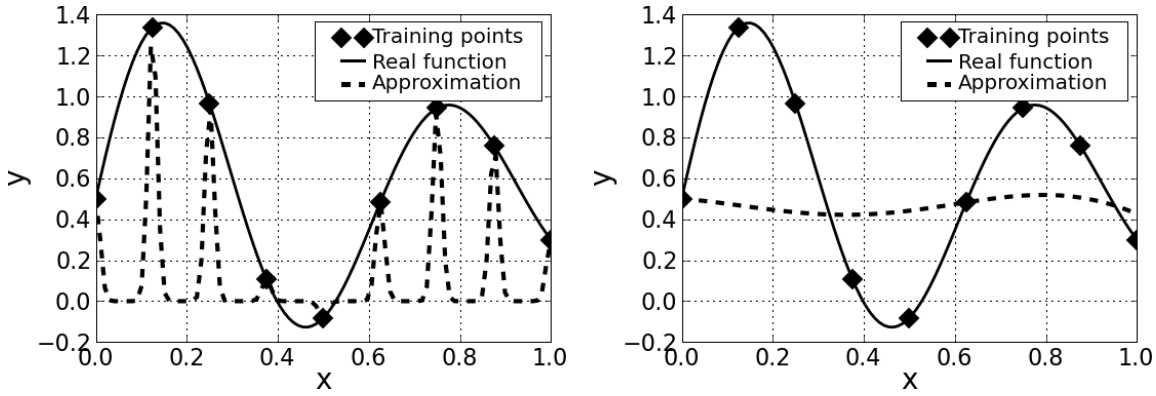
(b) AAE on experimental data

(c) Section through $C = 30$

(d) Best approximation with SVR

Figure 3.4: Finding optimal parameters

The function approximations are plotted for extreme values of σ in Fig. 3.5. These plots highlight the characteristics of a SVR approximation with incorrect σ values. The function approximation is seen to spike when sigma is too low ($\sigma = 0.01$) in Fig. 3.5a. In contrast, when the σ value is too high ($\sigma = 1$), the approximation tends towards a constant mean value as shown in Fig. 3.5b. It is therefore clear that the approximation error is subject to choosing the correct parameters.



(a) σ selected too low (= 0.01) (b) σ selected too high (= 1)

Figure 3.5: Incorrect σ values at $C = 30$

3.3.2 Investigating the effect of the GRBF kernel

When the GRBF kernel is used, the final function consists of a summation of the GRBF kernel at each training point. From the SVR optimization problem, two variables, namely an α and an α^*, are allocated to each training point. The coefficient $(\alpha^* - \alpha)$ in every function approximation indicates the weight of the GRBF around a training point. A single σ is used for all training points, and therefore all have the same radial weight. In Fig. 3.6 each individual GRBF can be seen, while the final approximation is the sum of every GRBF at each training point. When σ is very low, each training point will only have a small radial effect around itself. For this reason, the function approximation 'spikes' through the original training points at low σ values, and acquires a constant value at high σ values.

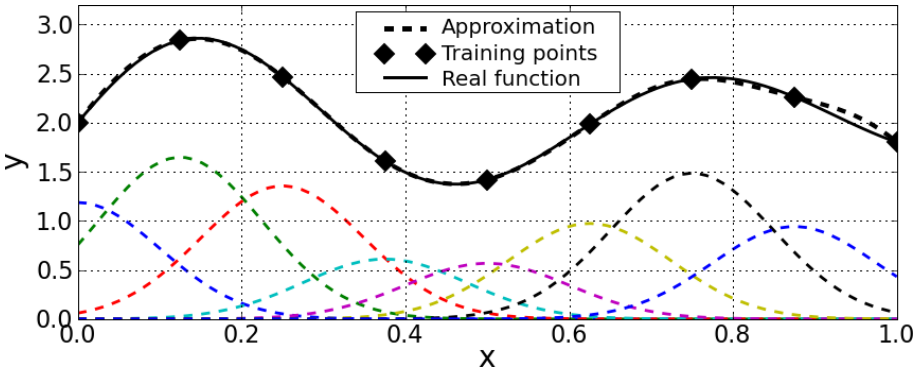


Figure 3.6: Effect of adding the individual GRBF functions

3.3.3 Using a different definition for ε

The value of ε can be defined as $\varepsilon = C \times 10^{-4}$ as in Gunn's [17] Matlab code . It is found that by using this definition, a specific range of C exists for an optimal regression. When C is increased past this point, the regression approximation becomes less accurate. In Fig. 3.7 the optimal areas for C and σ selection can be seen.

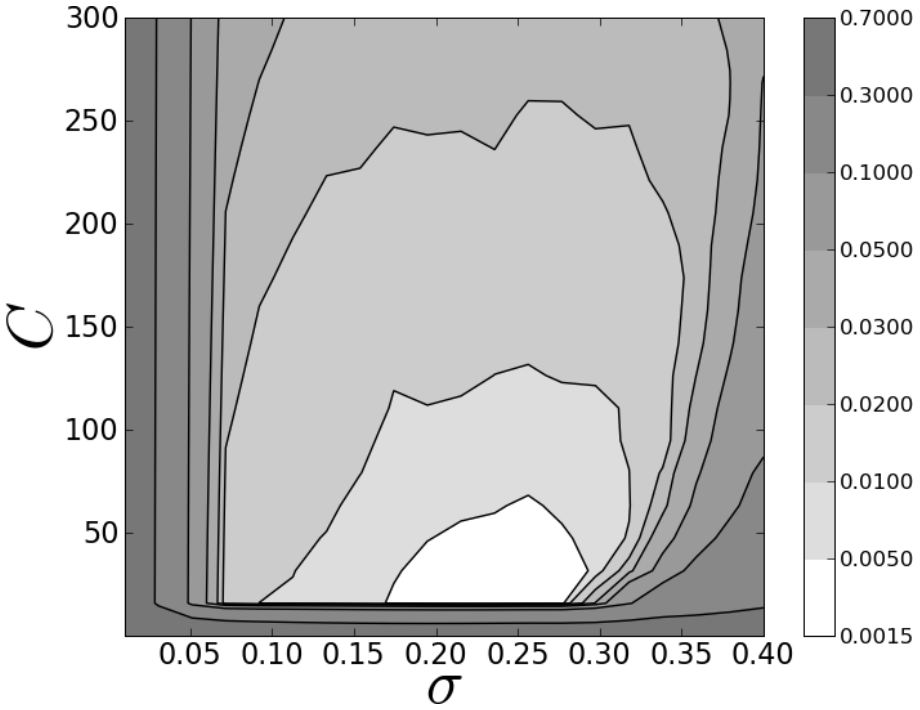


Figure 3.7: Approximation error on alternate definition for ε

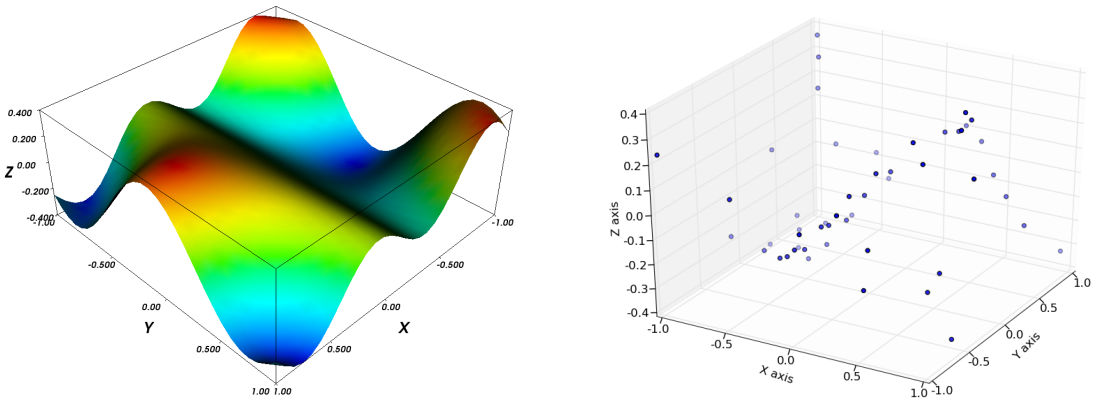
Defining ε in this way can serves two purposes. Firstly, the value ε is incremented to a value at which the noise of the original training data can be contained inside the ε corridor. Secondly, the parameter ε could be automatically defined. However, this technique was found to yield less accurate results for the applications in this thesis and is therefore not used further.

3.3.4 Approximating a function of two variables

A function of two variables is presented in this section. The function utilized is described in Eq. 3.33 below:

$$f(\mathbf{x}) = \sin(2x_1)\cos(4x_1x_2) \tag{3.33}$$

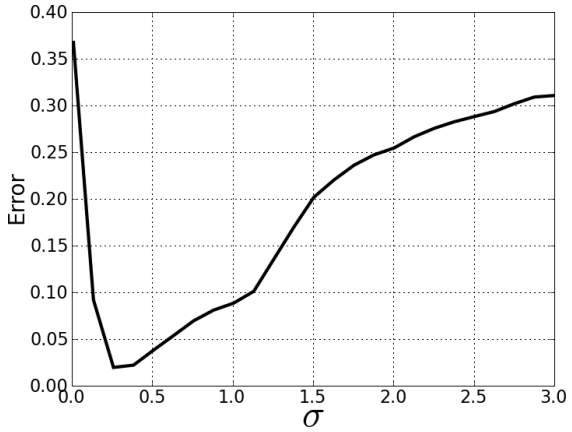
which is presented in Fig. 3.8a. This function is chosen because it has sufficiently non-linear characteristics to require a metamodelling technology like SVR. From this original function, 51 training points are randomly chosen to determine the effect of a random distribution, and are shown in 3.8b. The original function is then approximated with SVR using a GRB kernel. A separate data set of 400 evenly distributed points was created with the meshgrid function in Python to determine the AAE, as shown in Eq. 3.2.



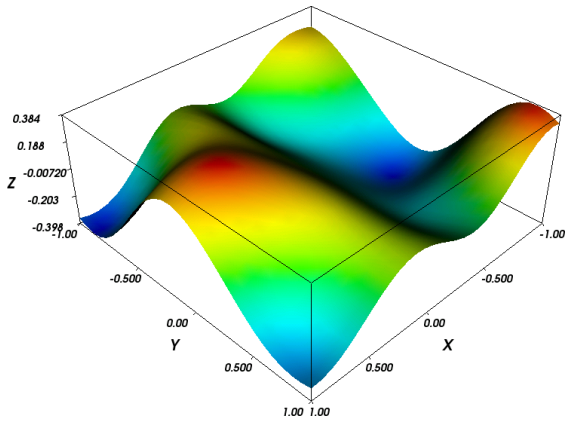
(a) Original function (b) Training points

Figure 3.8: Original function and training points

The value of C is taken to be 100, because an increase in this value of C does not increase the accuracy. The value of σ is then varied to determine the effect on the accuracy of the function approximation. The σ vs. error graph in Fig. 3.9a again shows the existence of a band in which an optimal σ value can be found. At this optimal value, the approximation has a small absolute-average-error, and is shown in Fig. 3.9b. At the optimal value, larger errors were found in the areas that the random distribution does not effectively represent.



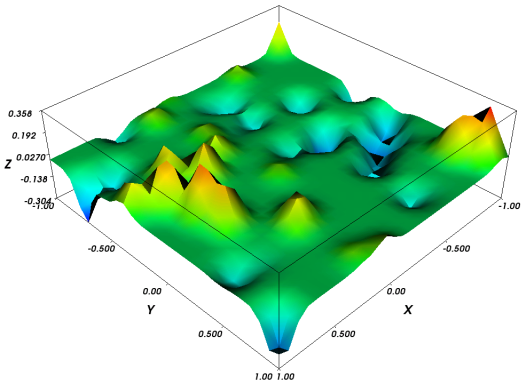
(a) Optimizing for σ



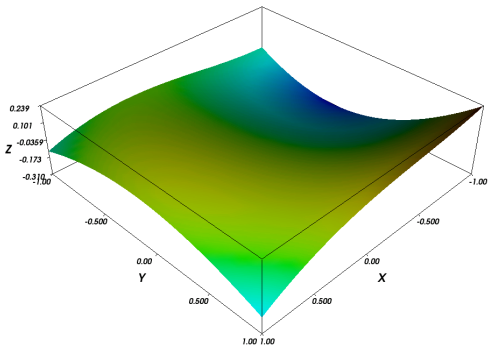
(b) Optimal regression fit at $\sigma = 0.4$

Figure 3.9: Three dimensional regression

It is seen again that when a σ is chosen too small, the approximation 'spikes' as shown in Fig. 3.10. When an excessively high σ is selected as shown in Fig. 3.10, the approximation again starts tending towards a constant value across the design space.



(a) σ too low



(b) σ too high

Figure 3.10: Using incorrect σ values

3.3.5 Discussion

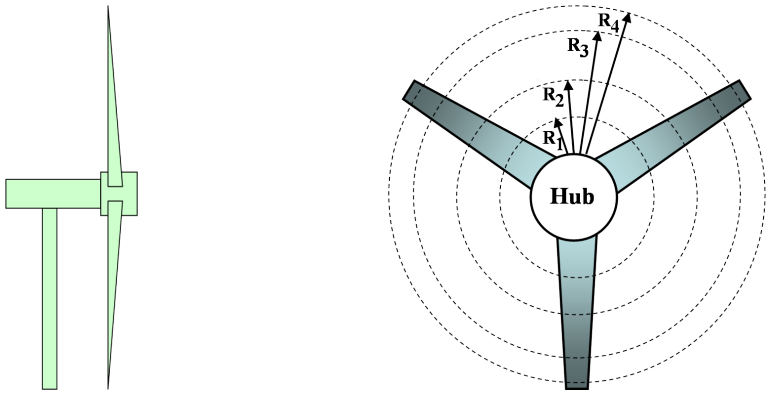
At low values of σ, the SVR approximation was found to form 'spike', which allowed it to fit exactly through all the original training points. It is seen that at high values of σ, the approximation tends towards the constant value of the mean of the training points. The approximation error is highly sensitive to changes in σ, but insensitive to changes

in C, providing C is sufficiently high, in the case of a fixed ε value, equal to 10^{-4}. It was also found that when insufficient points are utilized the minimum error becomes unacceptably high. In conclusion, sufficiently complex problems have been explored in this section to explore the analysis of problems in higher dimensions.

Chapter 4

Low-speed wind turbine theory

IN order to demonstrate the use of Support Vector Regression (SVR), an application with non-linear response characteristics was sought. The optimization of a low-speed Horizontal Axis Wind Turbine (HAWT), as shown in Fig. 4.1a, was decided upon as it has non-linear characteristics and the effect of SVR on higher dimensions can be explored.



(a) Layout (b) Four station approximation

Figure 4.1: Horizontal Axis Wind Turbine

The objective of this demonstration problem is to use the 4-digit NACA airfoil profiles to optimize a wind turbine. In this problem, the design variables are the airfoil geometries, chord lengths and pitch angles. Pitch angles define the angle at which the blade stations are rotated relative to the rotational plane of the turbine. Following the example of Cencelli [6], this problem makes use of four radial stations, as shown in Fig. 4.1b. Each blade station has its own NACA airfoil, pitch vector and chord length. This section

introduces the theory describing wind-turbines, as the theory had to be investigated before the turbine could effectively be optimized.

4.1 The NACA airfoil

The NACA 4-digit airfoil [18] has 4-digits, namely k, l and mm that describes the general geometry, and another variable c that describes the chord length. The camber (y_C) and thickness (y_T) functions describing the airfoil profile are calculated using the following equations:

$$y_C = \begin{cases} y_{Cmax}(2\lambda x_s - x_s^2)/\lambda^2 & \text{for} \quad 0 \leq (x_s) \leq \lambda \\ y_{Cmax}(1 - 2\lambda + 2\lambda x_s - x_s^2)/(1-\lambda)^2 & \text{for} \quad \lambda < (x_s) \leq 1 \end{cases} \tag{4.1}$$

$$y_T = \quad y_{Tmax}(1.48\sqrt{x_s} - 0.63x_s - 1.76x_s^2 + 1.4215x_s^3 - 0.51x_s^4) \tag{4.2}$$

where $x_s = x/c$ and x varies from 0 to the length of the chord. The value of y_{Cmax} represents the value of the maximum camber, y_{Tmax} the maximum thickness and λ specifies the location of the maximum camber. These three unknown values are derived from the k, l and mm values as follows:

$$y_{Cmax} = k \times c/100 \tag{4.3}$$

$$y_{Tmax} = mm \times c/100 \tag{4.4}$$

$$\lambda = l/10 \tag{4.5}$$

The final geometry is divided into an upper and lower surface, described as:

$$\text{Upper surface} \qquad y_U = y_C(x) + y_T(x) \tag{4.6}$$

$$\text{Lower surface} \qquad y_L = y_C(x) - y_T(x) \tag{4.7}$$

The geometry of a NACA4414 airfoil is shown in Fig. 4.2, where the x-axis represents x_s. By adjusting the $klmm$ values the thickness of the blade and the inflection point of the camber can be varied. It is now possible, as opposed to the majority of other airfoil types, to design a broad spectrum of airfoil geometries with a mathematical formula.

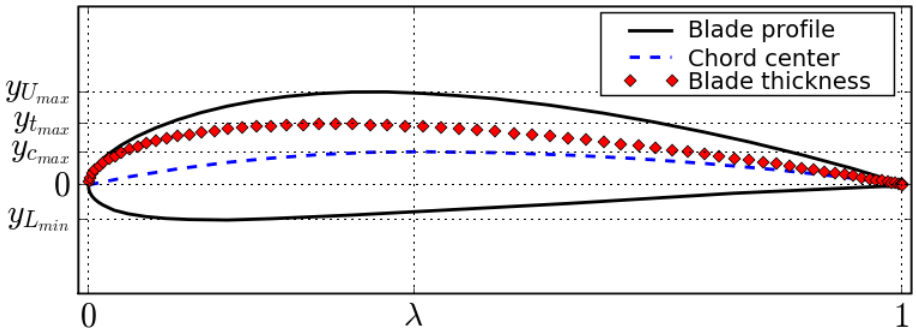


Figure 4.2: A NACA 4-digit airfoil

4.2 One dimensional momentum theory

A wind turbine cannot be approximated as a normal aircraft wing because of the vortices that are created in its wake as it rotates. The flow over a wind turbine can, however, be approximated as a permeable rotor or disc [3] as seen in Fig. 4.3. The disc approximates the turbine without the effects of friction and rotational velocity and acts as a drag device, slowing the wind down from V_o upstream to u at the blade and u_1 downstream. This section introduces axial and tangential induction factors to approximate the effect of the rotation.

4.2.1 Axial induction factor

In this section an approach is developed to approximate the effect that rotational velocities have on the axial velocity, namely an axial induction (a_a) factor. To begin, the pressure and axial velocity profiles are shown in Fig. 4.3. To find a_a, the definition for thrust and power is first derived from the ideal disc approximation. The thrust force (T) operates in the stream-wise direction, braking the wind-speed from V_o to u_1:

$$T = \Delta p A \tag{4.8}$$

where Δp is the pressure drop over the turbine and the area, A, is defined as:

$$A = \pi R^2 \tag{4.9}$$

with R the radius of the turbine. Bernoulli's equation [3] is valid for the flow over the turbine, and is calculated as follows :

$$\underbrace{p_o + \frac{1}{2}\rho V_0^2}_{\text{Upstream}} = \underbrace{p + \frac{1}{2}\rho u^2}_{\text{In front of rotor}} \tag{4.10}$$

where ρ is the air density, V_o is the approaching velocity, u is the velocity at the turbine, p_0 is the air-pressure far from the turbine and p is the air-pressure at the turbine. The flow downstream of the turbine is described as:

$$\underbrace{p - \Delta p + \frac{1}{2}\rho u^2}_{\text{Behind rotor}} = \underbrace{p_0 + \frac{1}{2}\rho u_1^2}_{\text{Downstream of rotor}} \tag{4.11}$$

By combining Eq.'s 4.10 and 4.11 an expression for the pressure drop is obtained:

$$\Delta p = \frac{1}{2}\rho(V_0^2 - u_1^2) \tag{4.12}$$

The axial momentum equation, described in Hansen [3], can be applied to the control volume shown in Fig. 4.3 as follows:

$$\frac{\partial}{\partial t}\int\int\int_{CV} \rho U d(vol) + \int\int_{CS} U\rho \boldsymbol{V} d\boldsymbol{A} = F_{ext} + F_{pres} \tag{4.13}$$

where $d\boldsymbol{A}$ is a vector pointing in the direction normal to of infinitesimal area of the control surface [3]. The variable U describes the velocity in the flow field. F_{pres} is the pressure force acting on the control volume and $d(vol)$ denotes a section of the control volume. Since the flow is assumed stationary, the first term in Eq. 4.13 is zero since the flow is stationary. The last term F_{pres} is zero because pressure on the end planes of the control volume is equal. Using these assumptions an ideal rotor equation for thrust can be derived as follows:

$$\rho u_1^2 A_1 + \rho V_0^2(A_{cv} - A_1) + \dot{m}_{side} V_0 - \rho V_0^2 A_{cv} = -T \tag{4.14}$$

where A_{cv} and A_1 are the areas of the control-volume and the area at the exit of the flow-field respectively, as seen in Fig 4.3. From the law of conservation of mass the flow that exits the control volume on the sides can be calculated as:

$$\dot{m}_{side} = \rho A_1(V_0 - u_1) \tag{4.15}$$

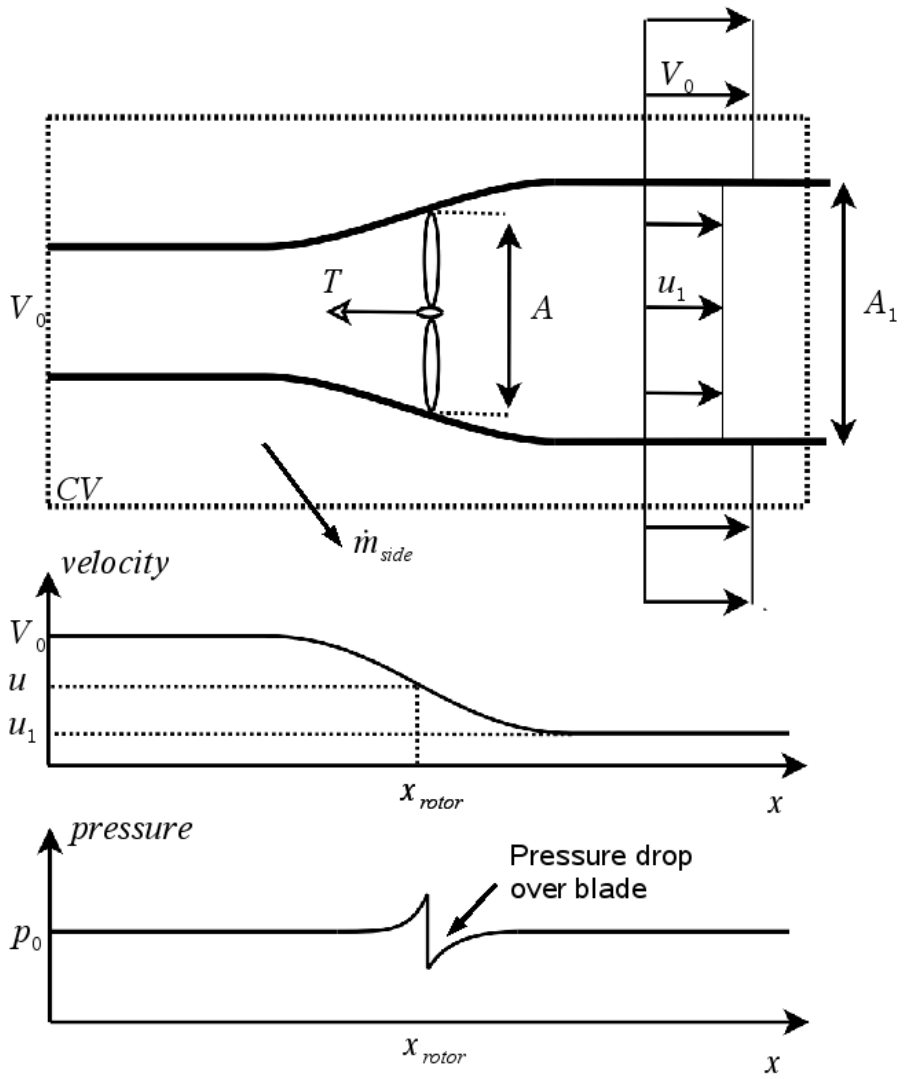


Figure 4.3: Control volume around wind turbine [3]

A relationship between the mass flow at the blade and at the exit of the control volume can also be derived from the conservation of mass as follows:

$$\dot{m} = \rho u A = \rho u_1 A_1 \tag{4.16}$$

When Eq.'s 4.14, 4.15 and 4.16 are combined, a new expression for thrust is obtained:

$$T = \rho u A(V_0 - u_1) = \dot{m}(V_0 - u_1) \tag{4.17}$$

Now the velocity at the blade, u, can be found by combining Eq. 4.17 with Eq. 4.8 as follows:

$$u = \frac{1}{2}(V_0 + u_1) \tag{4.18}$$

The rotor disk acts as a permeable membrane that retards the flow through it. The axial induction factor is introduced to approximate the velocity reduction over the turbine as:

$$u = (1 - a_a)V_0 \tag{4.19}$$

Combining Eq.'s 4.19 and 4.18 an equation for the upstream velocity is obtained:

$$u_1 = (1 - 2a_a)V_0 \tag{4.20}$$

4.2.2 Tangential induction factor and blade angles

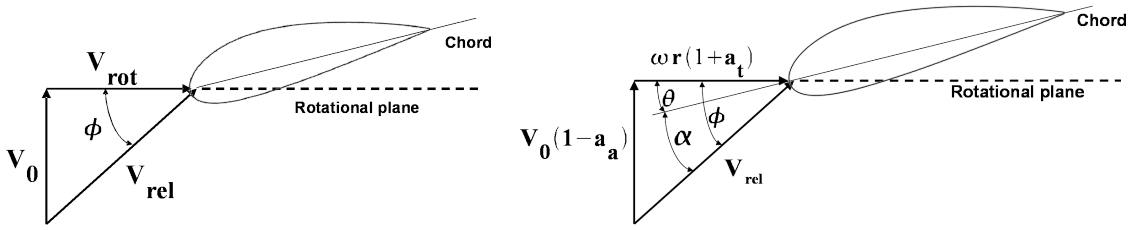
A tangential induction factor (a_t) also exists to account for the vortices in the rotational direction. The derivation is given in Hansen [3], where the induction factor effectively increments the rotational velocity. The new rotational velocity is now seen by the blade as:

$$V_{rot} = \omega r(1 + a_t) \tag{4.21}$$

where ω is measured in rpm and r is the radial distance to the location on the blade at which the evaluation is done. Finally, the relative velocity V_{rel} that is seen by the blade can be calculated as follows:

$$V_{rel} = \sqrt{V_{rot}^2 + u^2} \tag{4.22}$$

In Fig. 4.4 the effect of the change of velocities as a result of vortices and the consequent induction factors are shown.



(a) Velocities without vortices (b) Velocities with vortices [3]

Figure 4.4: Comparison of wind velocities

The pitch vector, θ, is the physical angle at which the blade is set relative to the rotational plane. The angle, ϕ, that V_{rel} makes to the rotational plane can be evaluated as:

$$tan\phi = \frac{(1 - a_a)V_o}{(1 + a_t)\omega r} \tag{4.23}$$

The local angle of attack can now be found as:

$$\alpha = \phi - \theta \tag{4.24}$$

as shown in Fig. 4.4b. The induction factors can now be used to approximate the wind angles in a wind-turbine.

4.2.3 Power coefficient from induction factors

The objective when optimizing a wind turbine is usually to maximize the power coefficient. However, the power coefficient C_p can only be found once the induction factors have been calculated. To find the power coefficient a number of steps have to be taken. Firstly, an expression for power [3] can be derived when a control volume is taken that is aligned with the streamlines in Fig. 4.3 as follows:

$$P = \dot{m}(\frac{1}{2}V_0^2 + \frac{p_0}{\rho} - \frac{1}{2}u_1 - \frac{p_0}{\rho}) \tag{4.25}$$

where ρ is the density of the air. Substituting Eq.4.16 into Eq.4.25, the expression for power is reduced to:

$$P = \frac{1}{2}\rho u A(V_0^2 - u_1^2) \tag{4.26}$$

Expressing u_1 as induction factors for Eq. 4.20 yields:

$$P = 2\rho V_0^3 a_a(1 - a_a^2)A \tag{4.27}$$

Finally, the power can be expressed as a ratio of the available power to total power extracted from the system:

$$C_p = \frac{P}{P_{avail}} \tag{4.28}$$

where the available power is:

$$P_{avail} = \frac{1}{2}\dot{m} = \frac{1}{2}\rho V_0^3 A \tag{4.29}$$

Since power is a function of the induction factors, the C_p can also be expressed in terms of the a_a by substituting Eq.'s 4.27 and 4.29 into Eq. 4.28:

$$C_p = 4a_a(1 - a_a)^2 \tag{4.30}$$

To find the maximum C_p, the derivative of Eq. 4.30 is taken:

$$\frac{\mathrm{d}C_p}{\mathrm{d}a} = 4(1-a)(1-3a) \tag{4.31}$$

and set equal to zero to find the optimum. The optimum is found at $a = 1/3$, yielding $C_p = 16/27$ which is known as the Betz limit.

4.3 Blade Element Momentum theory

The Blade Element Momentum (BEM) theory simplifies the simulation for a rotating blade by approximating the blade as a series of annular 2-D segments. A 2D profile has no depth and it is assumed the wing stretches into an infinite span. The primary assumption of the Blade Element Momentum theory is that blade station can be calculated separately, and is not influenced by the flow from neighboring blade stations. The BEM theory therefore assumes that the flow at each segment of blade is independent of flow at bordering segment.

4.3.1 Forces on blade

When the lift and drag coefficients are known for a blade, the lift and drag forces on the blade can be calculated as follows [3]:

$$F_L = \frac{c_l \rho V_{rel}^2 c}{2} \tag{4.32}$$

$$F_D = \frac{c_d \rho V_{rel}^2 c}{2} \tag{4.33}$$

where c_l is the lift coefficient, c_d is the drag coefficient, ρ is the density and c is the chord length. Figure 4.5 shows the forces being translated onto the plane parallel and perpendicular to the rotational plane. The normal and tangential forces can be calculated as:

$$F_N = F_L \cos\phi + F_D \sin\phi \tag{4.34}$$

$$F_T = F_L \sin\phi - F_D \cos\phi \tag{4.35}$$

and are calculated per unit length.

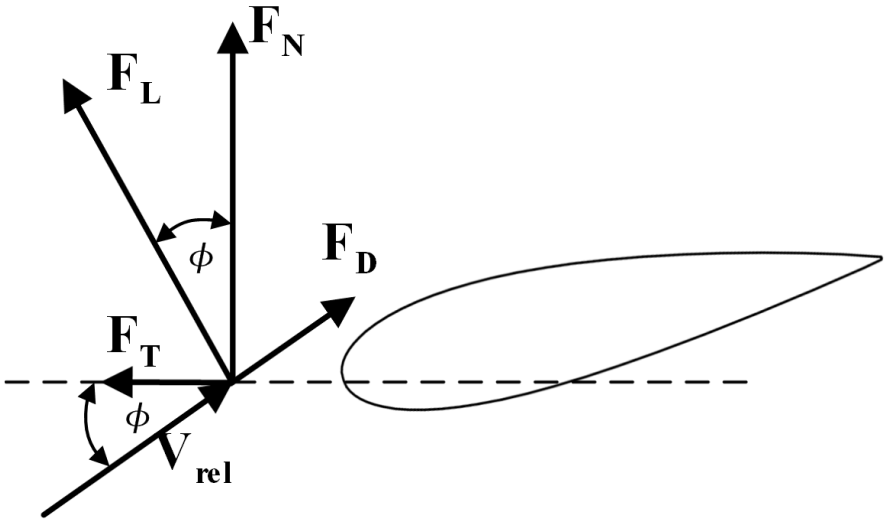


Figure 4.5: Forces on blade [3]

When these forces are scaled with respect to $\frac{1}{2}\rho V_{rel}^2 c$, the normal ($c_n$) and tangential ($c_t$) coefficients are found:

$$c_n = c_l \cos\phi + c_d \sin\phi \tag{4.36}$$

$$c_t = c_l \sin\phi - c_d \cos\phi \tag{4.37}$$

With the translated values and induction factors it is now possible to work on the rotational axis of the turbine and find the C_p.

4.3.2 Power coefficient from forces

The torque generated per radial section of the turbine across the entire radial area of the turbine can be calculated as [3]:

$$dM = rBF_T \mathrm{d}r \tag{4.38}$$

where r is the radius to the centre of the blade station, B is the number of blades and dr is the length of the blade segment. The power is calculated as the sum of the power generated at each station:

$$P = \sum dP = \sum \omega dM \tag{4.39}$$

where ω is the rotational velocity. Substituting Eq.'s 4.39 and 4.29 into Eq. 4.28, the power coefficient can now be determined as sum of the moments about the hub as

follows:

$$C_p = \frac{P}{\frac{1}{2}\rho V_o^3 A} = \frac{\sum \omega dM}{\frac{1}{2}\rho V_o^3 A} \tag{4.40}$$

which yields the torque for the entire blade. A similar methodology can be followed if the C_p per blade station needs to be calculated.

4.3.3 Development of induction factors

This section presents the equations that are derived in Hansen [3] to approximate the axial and tangential induction factors. The initial induction factors were found to be:

$$a_a = \frac{1}{(4\sin\phi^2/\sigma c_n + 1)} \tag{4.41}$$

$$a_t = \frac{1}{[4\sin\phi\cos\phi/\sigma c_t - 1]} \tag{4.42}$$

where σ is a value known as the solidity of the turbine. The solidity describes the fraction of area occupied by blade to space on a specified radius as follows:

$$\sigma(r) = \frac{c(r)B}{2\pi r} \tag{4.43}$$

The induction factors have undergone two modifications [19] since they were conceived, namely Prandtl's tip loss factor and Glauert's correction factor for high a_a values. These modifications are discussed briefly in the following section.

Prandtl's tip loss factor

A modification to Eq.'s 4.41 and 4.42 was made by Prandtl to account for tip losses [19] as follows:

$$a_a = \frac{1}{[4F\sin\phi^2/\sigma c_n + 1]} \tag{4.44}$$

$$a_t = \frac{1}{[4F\sin\phi\cos\phi/\sigma c_t - 1]} \tag{4.45}$$

where F is defined as

$$F = \frac{2}{\pi}\arccos(e^{-f}) \tag{4.46}$$

where

$$f = \frac{B}{2}\frac{R-r}{r\sin\phi} \tag{4.47}$$

These equations should be used instead of Eq.'s 4.41 and 4.42

Glauert's correction factor

It was found that the BEM theory breaks down at high values of a_a [3]. A variable a_c is assigned to the value at which the BEM breaks down, where $a_c \approx 0.4$, [19]. The Glauert correction factor is evaluated as follows:
If $a_a > a_c$:

$$a_a = \frac{1}{2}[2 + K(1 - 2a_c) - \sqrt{(K(1 - 2a_c) + 2)^2 + 4(Ka_c - 1)}] \tag{4.48}$$

where

$$K = \frac{4F \sin\phi^2}{\sigma c_n} \tag{4.49}$$

otherwise Eq. 4.44 must be used. The tangential induction factor given in Eq. 4.45 is for all values of a_a.

4.4 BEM Algorithm

The induction factors need to be found by an iterative procedure, since the c_l and c_d values depend partially on the induction factors, and the induction factors depend on the values of c_l and c_d. The algorithm used in this thesis is a variation on the algorithm described in Hansen [3], and is presented in Table 4.1.

Table 4.1: BEM algorithm

Step 1.	Initialize a_a and a_t to 0
Step 2.	Compute the flow angle ϕ using Eq.4.23
Step 3.	Compute the local angle of attack with Eq.4.24
Step 4.	Compute $c_l(\alpha)$ and $c_d(\alpha)$ from SVR approximation
Step 5.	Compute F_N and F_T from Eq.4.34 and Eq.4.35
Step 6.	Calculate a_a and a_t from Eq.4.44, Eq.4.48 and Eq.4.45
Step 7.	Iterate from Step 2 until C_p converges

Step 1 in Table 4.1 shows the initialization of the induction factors. The accuracy at which convergence is assumed for C_p is defined as follows:

$$\frac{C_p - C_{pold}}{C_p} \leq 10^{-5} \tag{4.50}$$

The C_{pold} value is the C_p value from the prior iteration. As shown in step 1, the algorithm initializes the induction factors to 0 on the first function call from the optimizer. The reason for the small convergence value is to reduce numerical noise in the C_p function during an optimization procedure, and by reducing the convergence value the noise is reduced to a degree.

Once a set of induction factors are available for a specific optimization routine, the values are used to initialize the next design point called by the optimizer. Therefore, each subsequent function call retrieves the converged induction factors of the previous function evaluation, thereby reducing the number of iterations required in an optimization. Step 4 shows the c_l and c_d being found from a SVR metamodel, as opposed to evaluating the coefficients with a software simulation. Using a metamodel instead of directly coupling the BEM theory to XFOIL to evaluate the lift and drag significantly reduces the time to evaluate the power coefficient. With the method discussed in this chapter, it is now possible to evaluate the power coefficient of a wind turbine.

Chapter 5

Method of optimization

TO optimize the wind-turbine, a number of steps need to be taken. This section describes the integration of the Blade Element Momentum theory and the Support Vector Regression metamodelling technology, as well as the methodology followed in the optimization process.

5.1 Blade representation

In utilizing the Blade Element Momentum (BEM) theory, each turbine blade is represented by four radial stations. The four stations can be described as the root, semi, mid and tip station respectively when moving radially outward from the hub. The centres of the radial stations are taken at the following four locations:

$$r = [0.2,\ 0.5,\ 0.75,\ 0.95] \times R \tag{5.1}$$

where R represents the design blade radius. Figure 5.1 shows a typical representation of a four-station blade, in which each station has an independent profile, pitch angle and chord length.

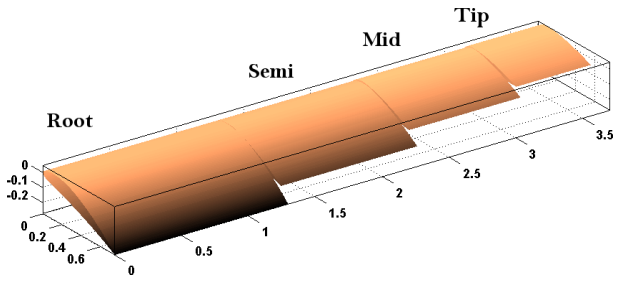


Figure 5.1: Graphical representation of approximation

5.2 Defining design variables

The fraction of power that can be drawn from the wind with an unshrouded wind turbine is 16/27, also known as the Betz limit [19]. The power that can be extracted from the wind with a Horizontal Axis Wind Turbine (HAWT) is dependant on a number of variables. These variables can be split into two sections, the fixed or constant design parameters, and the design variables. In this optimization problem, the design parameters essentially describe the environment in which the wind turbine is to be optimized. The design parameters used here are similar to the ones used by Cencelli [6] and are shown in Table 5.2.

Table 5.1: Design parameters

Variable	Symbol	Values assigned
Radius	R	$3.7m$
Rotational velocity	ω	$80rpm$
Axial velocity	V_o	$5-7m/s$
Density	ρ	1.0303
Viscosity	υ	$15.69e-06$

The density and viscosity both describe the air at Pretoria, Gauteng, South Africa, at an altitude of 1370m above sea level. An average incoming wind speed of $V_o = 6m/s$ is designed for, at a turbine rotation of $80rpm$. Lastly, a radius had to be designed for. Hansen [3] showed that a higher tipspeed ratio allows the design to attain a higher percentage of the Betz limit. Tipspeed ratio (λ) is defined as:

$$\lambda = \frac{\omega R}{V_o} \tag{5.2}$$

Choosing a higher tipspeed ratio increases the diameter of the turbine. Choosing a lower tipspeed means that a lower percentage of the Betz limit can be attained. A final tip speed ratio of 5 was chosen, as it allows a sufficiently high percentage ($\approx$ 97%) of

the Betz limit [3] to be attained. Substituting these variables into Eq. 5.2 above, a turbine radius of $R = 3.7$ is acquired. All the design parameters have now been defined.

5.3 Training points

5.3.1 Defining training points

The airfoil at a typical blade station can be described by the five design variables k, l, mm, θ and c. In the four-station approximation, the number of design variables is equal to 20, where values of k, l, mm, θ and c are allocated to each of the four stations. However, to evaluate the lift (c_l) and drag (c_d) coefficients at a blade station, the values of the five variables k, l, mm, α and Re are required by the simulation program, XFOIL. These five variables are used as training points. The variable α denotes the angle of attack and Re the Reynolds number. The difference between training points (TP's) and design variables (DV's) is shown in Table 5.2.

Table 5.2: Difference between TP's and DV's

TP	Relationship	DV
k	-	k
l	-	l
mm	-	mm
α	$\alpha = \phi - \theta$	θ
Re	$Re = \frac{V_{rel}c}{\upsilon}$	c

The principal reason for the difference in training points and design variables is that XFOIL requires the angle of attack, α, to determine c_l and c_d. However, the α value cannot be determined until the induction factors a_a and a_t are known. As a result, the pitch vector θ cannot be used for training data because it contains insufficient information to determine c_l and c_d.

The other changed variable is the value of V_{rel}, which represents the combined effect of the rotational and axial velocity, V_{rot} and V_o respectively (Eq. 4.22). The reason for the change from c to Re is that two variables, c and V_{rel} are combined, thereby reducing the dimensionality of the training problem. In training the SVR approximation, Re numbers are chosen in the range 50×10^3 to 2×10^6. This range covers all feasible Re number that will be required by the BEM theory for a turbine of this wind-speed. The

SVR approximation is trained with a suitable number of TP's, while the optimization procedure optimizes the DV's for each of the four stations.

5.3.2 Setting boundaries

In order to create a regression approximation, a number of training points need to be created, and need to be evaluated for the lift and drag coefficients, c_l and c_d. The training points also need to be evaluated in a feasible region. The bounds on the training points are chosen as seen in Table 5.3.

Table 5.3: The bounds of the training points

Variable	Lower limit	Upper limit
k	0	9
l	0	9
mm	0	20
α	0	20
Re	5×10^4	2×10^6

These bounds cover a large design space in which the optimal NACA airfoils are likely to be found. The training points were generated with the commercial code [1] called VisualDOT using the Design Of Experiments (DOE) program. Since no underlying function is presumed in the SVR approximation, it is advantageous to have the data dispersed optimally across the design space. The Optimal Latin hypercube design was chosen because of its space filling characteristics.

A sufficient number of training points were required to train the metamodel. The more points used result in a longer time to train the metamodel, while fewer points cause a lower accuracy. After a number of trial runs, a total of 600 training points were found to meet the balance between accuracy and time. Each point was evaluated using XFOIL [7] to obtain c_l and c_d. A number of the simulations produced no results, particularly at high α values because XFOIL failed to converge. A total of 315 training points were successfully evaluated, and stored in the format shown in Table 5.4.

Table 5.4: Table of inputs and outputs

$\boldsymbol{k}$	$\boldsymbol{l}$	$\boldsymbol{mm}$	$\boldsymbol{\alpha}$	$\boldsymbol{Re}$	$\boldsymbol{c_l}$	$\boldsymbol{c_d}$
k_1	l_1	mm_1	α_1	Re_1	$c_{l(1)}$	$c_{d(1)}$
k_2	l_2	mm_2	α_2	Re_2	$c_{l(2)}$	$c_{d(2)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
k_{n-1}	l_{n-1}	mm_{n-1}	α_{n-1}	Re_{n-1}	$c_{l(n-1)}$	$c_{d(n-1)}$
k_n	l_n	mm_n	α_n	Re_n	$c_{l(n)}$	$c_{d(n)}$

5.4 Optimization problem

The optimization of a wind turbine can also be mathematically represented as an objective function subject to certain constraints. The optimization problem is introduced here in the format described in Eq.'s 2.1 to 2.5 as follows:

$$\text{Maximize :} \quad C_p(\boldsymbol{k},\ \boldsymbol{l},\ \boldsymbol{mm},\ \boldsymbol{\theta},\ \boldsymbol{c}) \tag{5.3}$$

$$\text{Subject to :} \quad \alpha \leq 20 \tag{5.4}$$

$$\alpha \geq 0 \tag{5.5}$$

The C_p value is a function of the design point, ($\boldsymbol{k},\ \boldsymbol{l},\ \boldsymbol{mm},\ \boldsymbol{\theta},\ \boldsymbol{c}$), where the values $\boldsymbol{k}$ through $\boldsymbol{c}$ are four-element vectors describing the four radial stations. The objective function is to maximize the C_p value, or minimize the negative of C_p. Equations 5.4 and 5.5 constrain the angle of attack to between 0 and 20 degrees. The constraints on the angles of attack do not bound the final design, but serve to guide the optimizer into a feasible area faster in the initial iterations. No stress, displacement or frequency constraints are imposed to allow the optimizer the greatest degree of freedom in finding the optimum. However, by setting up the optimizer to optimize the blade in this manner, the designer has the freedom to impose these constraints easily. These constraints can be imposed, for example, by specifying decreasing chord lengths or decreasing thickness.

The bounds on the design variables, or side constraints, are shown in Table 5.5. The difference between the training points and design variables is outlined in Table 5.2. Usually the k, l and mm values are used as positive single digit integers, but by using the explicit equation describing the NACA 4-digit series the values can be chosen from all real numbers within specified bounds. The k, l values denote digits in a typical NACA$klmm$ airfoil, and are bounded between 0 and 9. The mm denotes a double digit range from 00 to 99, and is bounded between 5 and 20. The bounds on mm are chosen to prevent excessively thin or thick airfoils, as the airfoils starts acquiring infeasible

aerodynamic properties beyond these values.

Table 5.5: The bounds of the design variables

Variable	Lower limit	Upper limit
$\boldsymbol{k}$	0	9
$\boldsymbol{l}$	0	9
$\boldsymbol{mm}$	5	20
$\boldsymbol{\theta}$	0	20
$\boldsymbol{c}$	0.2	0.8

5.5 Methodology

This section gives an overview of the methodology followed to optimize the power coefficient of the wind turbine. Various programs were implemented in Python [20] to interface the various facets of the optimization routine.

5.5.1 Creation of the SVR metamodel

To create a SVR metamodel of the NACA 4-digit range, a total of 600 training points were generated and evaluated with XFOIL. The c_l and c_d at 315 of the points were successfully evaluated and used to create the SVR approximation. Each point was then scaled to values between 0 and 1. Another 500 separate data points were created with an OLH to be used as experimental or verification data. A total of 252 points were successfully evaluated with XFOIL. The experimental points are then used to determine the accuracy of the SVR metamodel at different parameter values.

The QP optimizer written by Vanderbei [13] is used to create the SVR lift and drag metamodels from the training data. A Gaussian Radial Base kernel was used, and so two parameters, C and σ, have to be chosen before the QP optimizer can be used to create the metamodel. The time required by the QP solver to create a regression of either the lift or drag function with the 315 points is $\approx$ 200 seconds on a 2.2 GHz 64 bit Dual Core AMD processor.

5.5.2 Creation of the RS metamodel

The same training points used for the SVR approximation were also used to create a 2^{nd} order polynomial, or Response Surface (RS), approximation of the lift and drag coefficients of the NACA 4-digit range. The RS metamodel is created to see whether the use of a more complicated metamodelling technology such as SVR is warranted. The time to approximate the 2^{nd} order polynomial approximation of either the lift or drag function requires roughly 1.5 seconds on the 2.2 GHz 64 bit Dual Core AMD processor.

5.5.3 Overview of optimizing C_p

The various facets of the C_p optimization are now ready to be implemented. In Fig. 5.2 a flowchart of the general procedure is shown. The left-hand side principally shows the procedure followed to create the metamodel, while the right-hand side of the flow-chart describes the optimization of the wind turbine. In the creation of the metamodel, 600 training points evaluated with XFOIL, with only 315 being successfully evaluated. The 315 points were then scaled and the optimal parameters for a SVR metamodel through these points was found. The resulting metamodels for c_l and c_d are then stored for later use by the BEM program.

For the optimization procedure, a total of 30 initial design points (***DP***'s) were defined with a DOE procedure to check whether local minima exist. The DOE procedure was created with an Optimal Latin Hypercube design. The optimizer started from each of the initial design points to optimize for C_p. The optimizer used the techniques described in section 2.1. For each gradient and function evaluation of the objective function C_p, the BEM program used the approximation created with SVR to find the relevant lift and drag coefficients, c_l and c_d. The optimizer finds the optimum C_p value for each initial design and stores the result.

An alternative metamodel was created with 2^{nd} order RS approximation. The RS metamodel is used in the same way as the SVR metamodel, as shown in Fig. 5.3. With this methodology it is possible to compare the results of the different metamodels and optimization techniques.

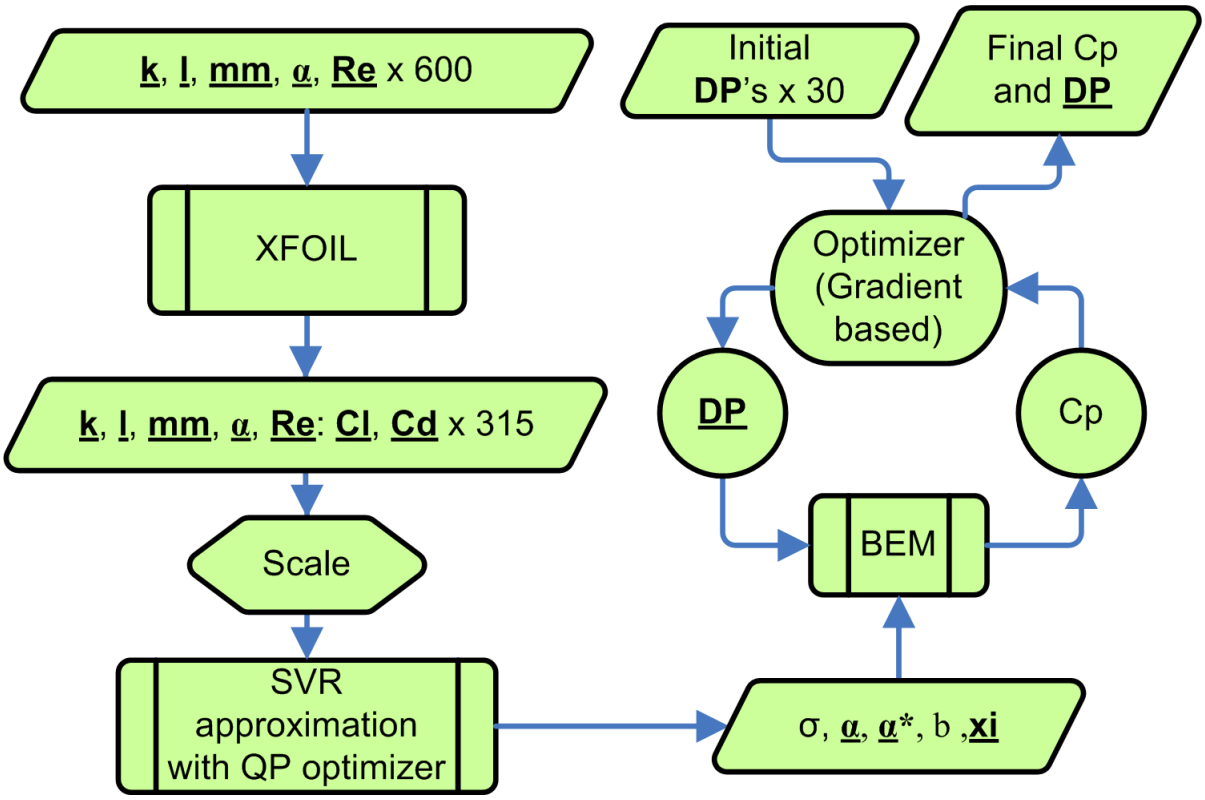


Figure 5.2: Flowchart of optimization methodology

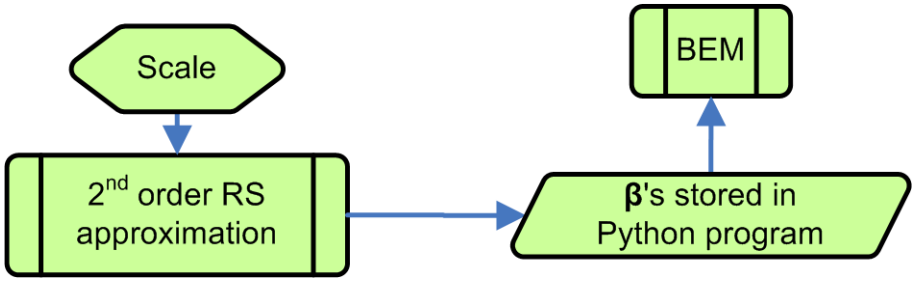


Figure 5.3: Using the RS metamodel

5.5.4 Convergence of C_p

Two factors have to taken into account when optimizing the C_p. The first of these is the number of divisions the wind-speed has to be divided into. The closer C_p is to convergence, the longer it takes to evaluate a C_p value at a design point. If there are 4 radial stations on the blade, the number of c_l and c_d evaluations per iteration in the BEM program can be calculated as:

$$n_{evals} = 4 \times n_{divs} \tag{5.6}$$

where n_{divs} is the number of wind divisions. If there are 5 wind-speed divisions for example, the wind-speed will divided up as follows:

$$\text{Wind-speeds} = [5,\ 5.5,\ 6,\ 6.5,\ 7] \tag{5.7}$$

In Fig. 5.4 the C_p can be seen to converge for an increase in increasing divisions. The change in the C_p with change in number of divisions is not significant, but it is enough to cause numerical problems in the optimization procedure.

Another factor to consider is the number of iterations in the BEM until the induction factors converge. In the Python program that implements the BEM algorithm, a tolerance value determines the number of iterations, as described in section 4.4. However, the number of iterations decisively affect the accuracy of the resultant C_p, as can be seen in Fig. 5.5 with 11 wind-speed divisions. Figure 5.4 has the tolerance on the C_p as described in section 4.4, while Fig. 5.5 is created with 11 divisions. The final optimization also makes use of 11 divisions, and the BEM program is run iteratively until the convergence criteria for C_p in Table 4.1 criteria is met.

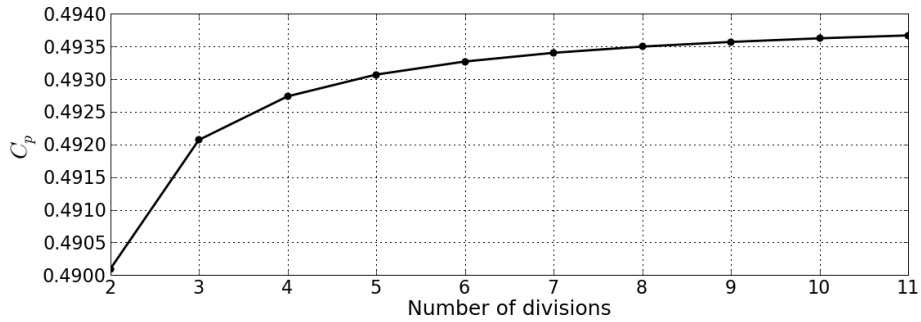


Figure 5.4: Converging C_p for increasing divisions at $tol \leq 10^{-5}$

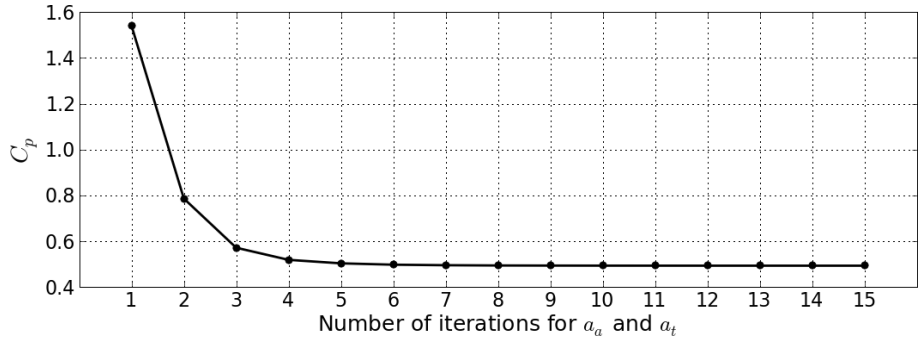


Figure 5.5: Converging C_p for increasing iterations with 11 divisions

5.5.5 Evaluation and verification with XFOIL

Once the optimal design point was found using the SVR or RSA approximations for the c_l and c_d values, it was necessary to verify the accuracy of the approximate optimum. The design point with a suitably high C_p value was chosen from the stored values. To verify the accuracy of the SVR approximation, this optimal design point was evaluated with the BEM program, but using XFOIL instead of using the SVR approximation to evaluate the lift and drag coefficients of the airfoils. The BEM program is therefore an interface program that can use different kinds of approximations, namely RS, SVR or XFOIL to evaluate the C_p value at a specified design point.

5.6 Conclusion

In conclusion, metamodels are created with SVR and RS for the c_l and c_d functions for the NACA 4-digit series within specified bounds. In approximating the C_p, all four stations make use of the same metamodels because all four stations make use of the NACA 4-digit series. Since the design wind speed is divided into 11 divisions and 4 stations are used, a total of 44 c_l values and 44 c_d values are evaluated in each iteration of the BEM program to evaluate the C_p value. This method of having only one metamodel for the c_l and another for the c_d NACA 4-digit series makes it very simple to increase the number of stations to improve the accuracy of the BEM approximation. However, the work in this thesis is limited to using only four stations.

Chapter 6

Results

THIS chapter presents the results obtained for the lift and drag metamodels created with Support Vector Regression (SVR) and Response Surface (RS) for the NACA 4-digit range. These metamodels were then used in conjunction with the BEM theory to optimize a low-speed wind-turbine. The results obtained for the creation of the lift and drag metamodels and optimization of the wind turbine are discussed.

6.1 Creating the c_l and c_d metamodels

The results of creating a metamodel for lift and drag coefficients, c_l and c_d, of the NACA 4-digit range is discussed here. The metamodels are created with ε-insensitive SVR using a Gaussian Radial Basis (GRB) kernel. An SVR approximation with a GRB kernel requires the adjustment of two variables, namely σ and C. In order to create the best approximation, a two layer optimization problem exists, with the outer loop optimizing the σ and C values, and the inner loop optimizing the α's and α^*'s in the SVR optimization problem (Eq. 3.20). The inner loop is a Quadratic Problem (QP) and the optimal α's are found with the QP solver written by Vanderbei [13].

6.1.1 Training points

A total of 600 data points were generated with an Optimum Latin Hypercube distribution for training the lift and drag SVR metamodels. After being evaluated with XFOIL, a total of 314 Original Data (OD) points were available for training. The OD points are then scaled between 0 and 1. To determine the accuracy of the parameters chosen for

the outer loop, a total of 500 Experimental Data (ED) points were created with a random distribution within the same bounds as the training points. A random distribution was used to evaluate the design at points away from the training points, and therefore an OLH is not necessary. The data points were then evaluated with XFOIL, with 252 points successfully evaluated and used. The reason for the high failure rate is due to the high angles of attack, low Reynolds numbers and infeasible airfoil shapes. Following the evaluation of the points with XFOIL, the ED points are then scaled according to the scale values used for the OD points. Once all the points have been scaled, the SVR approximations are created and the C_p values are optimized with the BEM theory.

6.1.2 Optimizing parameters for lift with SVR

With the set of training points for the lift coefficient, c_l, a SVR approximation is created. The two variables, C and σ, are varied to determine the accuracy of the SVR approximation at the varied values, as shown in Fig. 6.1. In this figure, the contours of the Absolute Average Error (AAE) are shown between the ED points and the SVR approximation with respect to varied C and σ values. A 'valley' exists in the error plot in which the ED error is below 2.4%. The σ and C values are chosen from within this area.

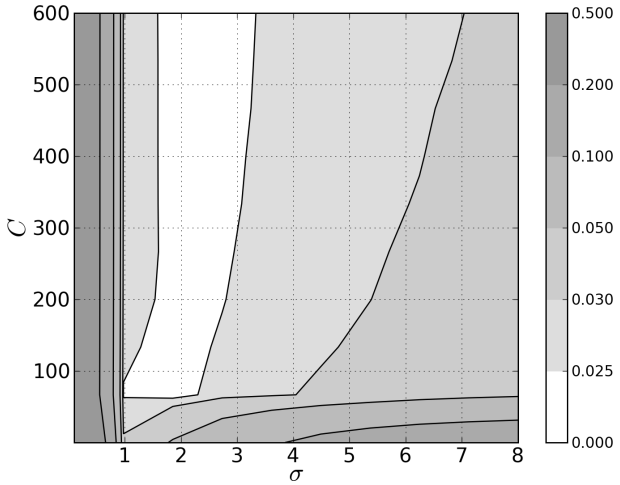


Figure 6.1: AAE between $\hat{c}_l$(**DP**) and real c_l with $\varepsilon = 10^{-4}$

A section plot is created through $C = 200$ is shown in Fig. 6.2 to more closely investigate the error response in the feasible section. In this graph the experimental error is the AAE between the SVR approximation and the experimental data. The original error is the AAE between the original training points and the SVR approximation at the corresponding σ values.

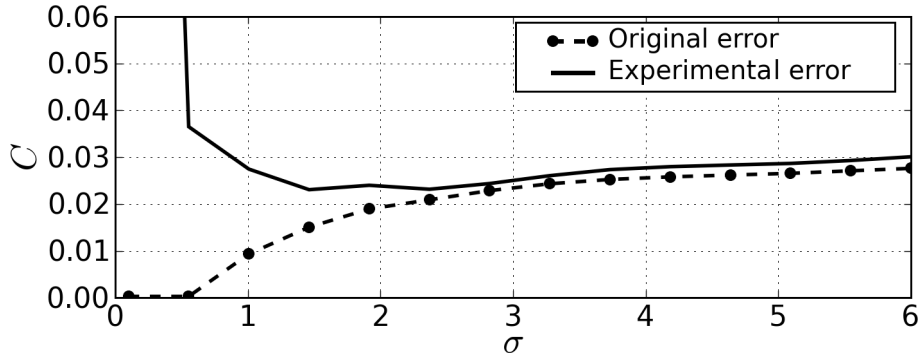


Figure 6.2: AAE at $C = 200$

At low values of σ, the Absolute Average Error (AAE) on the OD is smaller than ε. In the area where AAE $\leq \varepsilon = 10^{-4}$, the approximation interpolates all the original data points with a ε error. However, the minimum AAE of the ED is at $\sigma \approx 1.8$, where the AAE of the OD lies outside of the ε band. An error between the OD and the approximation greater than ε indicates that the approximation is no longer interpolating the OD points with a ε error.

The smallest AAE between the approximation and the ED occurs where the OD vs. approximation error is greater than ε. The fact that the optimal σ occurs where the OD error is greater than ε can be attributed to two possible factors. Firstly, it is possible that XFOIL has noise in the form of numerical error in its simulations. Secondly, it is possible that an insufficient number of training points are used in creating the approximation. At values of σ that are too low the SVR function approximation 'spikes' through or interpolates the training points. As a result, at low σ values no error is seen on the training data but a large error exists for the ED. At high values the approximation acquires a constant value (Section 3.3.1). The ED is therefore used to find the location of the best σ value. The final parameters for σ and C chosen for the final c_l approximation are chosen where the minimum ED error occurs. Since the C value only needs to be sufficiently high, the σ and C values are chosen at 1.8 and 200 respectively.

6.1.3 Optimizing parameters for drag with SVR

The optimal parameters for the approximation of the drag coefficient, c_d, is found following the same procedure outlined for the lift coefficient. The AAE between the SVR approximation and the ED is found by varying the σ and C values, as shown in Fig. 6.3. Again a small band again exists at which an optimal σ values are found at a sufficiently high C value. A cross section through Fig. 6.3 at $C = 200$ is shown in Fig. 6.4. The

lowest AAE is found at $\sigma = 1$ with an AAE of 0.018.

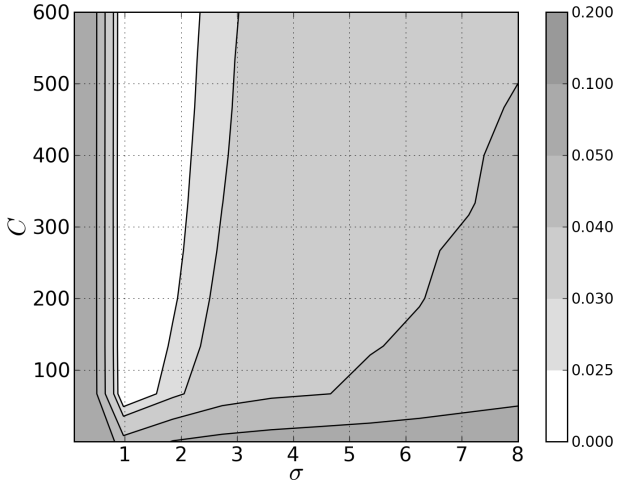


Figure 6.3: AAE between $\hat{c}_d$(**DP**) and $c_d$with $\varepsilon = 10^{-4}$

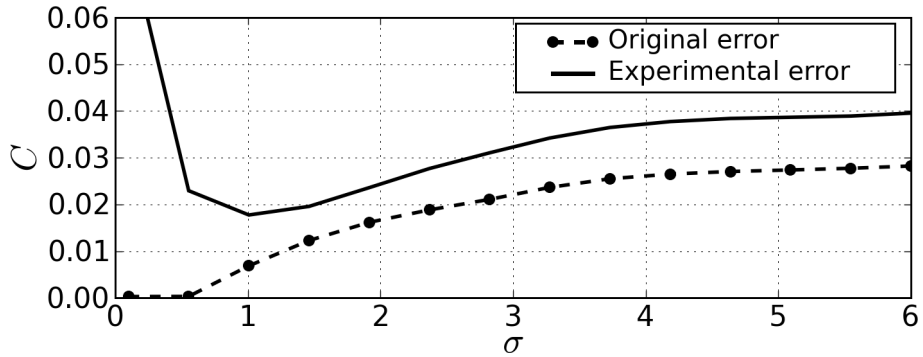


Figure 6.4: AAE at $C = 200$

Once the smallest errors have been found, all the information describing the SVR approximation is stored in memory for later use when optimizing the wind-turbine. The data that is stored includes the α's and α^*'s, the kernel parameter σ, the bias b and all the training points. The approximations can now be used instead of XFOIL to determine the lift and drag coefficients for all 4-digit NACA airfoil within the design boundaries.

6.1.4 Creating lift and drag metamodels with RS

A 2^{nd} order response surface metamodel is also used to create lift and drag metamodels. In creating these metamodels, the least-squares error method is used to find the best approximation, as discussed in section 2.2.4. As the name '2^{nd} order response' suggests, the lift and drag approximations assume quadratic functions. The metamodels

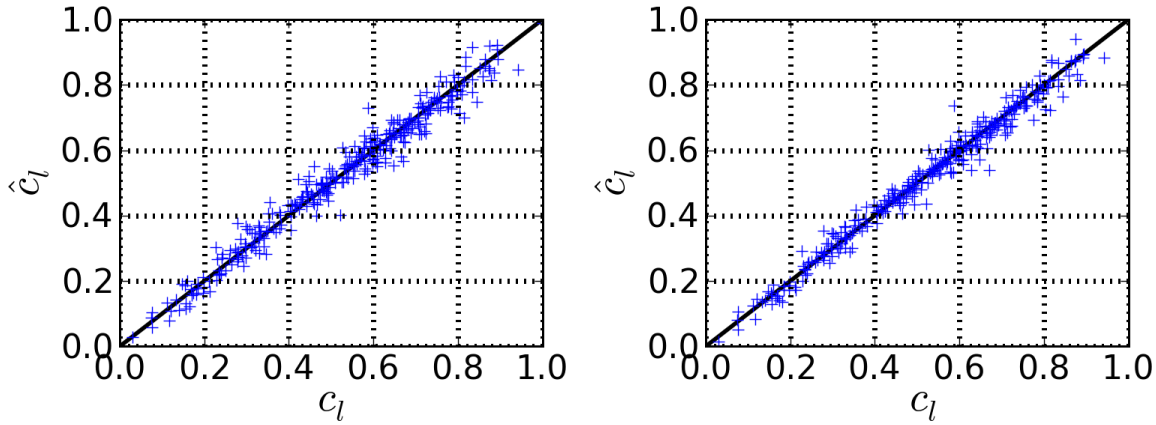
are created with the same 314 OD points used for the SVR metamodels. The AAE values found when comparing the ED to the metamodels are shown in Table 6.1. The lift error found with RS compares well with the best error found with SVR, while the RS drag approximation has more than double the error found with the SVR approximation.

Table 6.1: Accuracy of metamodels

Model	RS AAE	SVR AAE
Lift (c_l)	0.027	0.023
Drag (c_d)	0.043	0.018

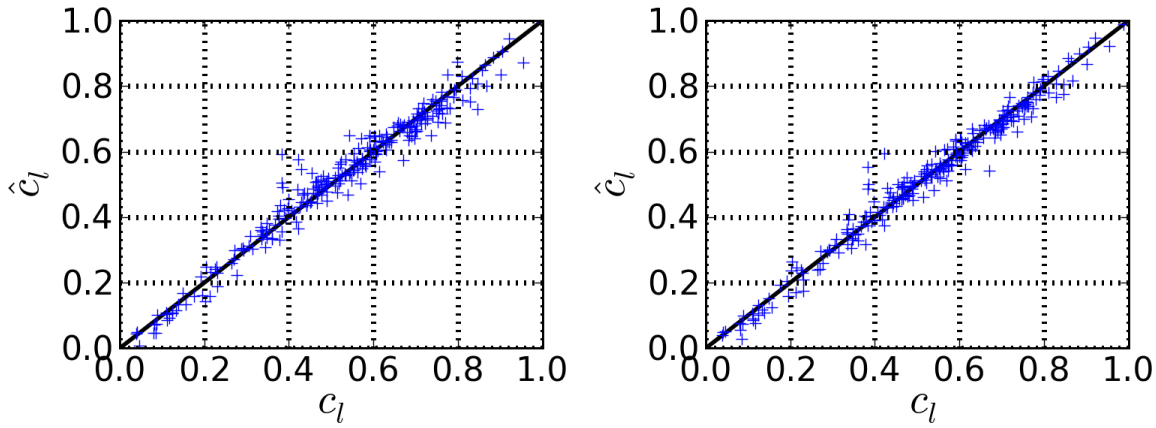
6.1.5 Lift comparison

In this section, the c_l approximations are compared to the simulated results obtained with XFOIL. Figures 6.5a and 6.5b show the RS and the SVR lift approximation respectively in comparison to the original training data (OD) obtained with XFOIL. The metamodels are also plotted against the experimental or verification data (ED) obtained with XFOIL in Fig.'s 6.6a and 6.6b. The accuracy of the SVR model has a similar accuracy between the OD and ED , which corresponds well with the SVR c_l error graph in section 6.1.2. The error between the OD and ED of the RS lift metamodel is also similar, indicating that sufficient training points were used for the RS metamodel. Furthermore, similar accuracies exist between the RS and SVR lift approximations, which means that the use of a complicated metamodelling technology such as SVR is probably not warranted in modelling the lift, as the lift seems to follow a second order polynomial trend.



(a) RS approximation on original data (b) SVR approximation on original data

Figure 6.5: Comparison of c_l metamodels on original data

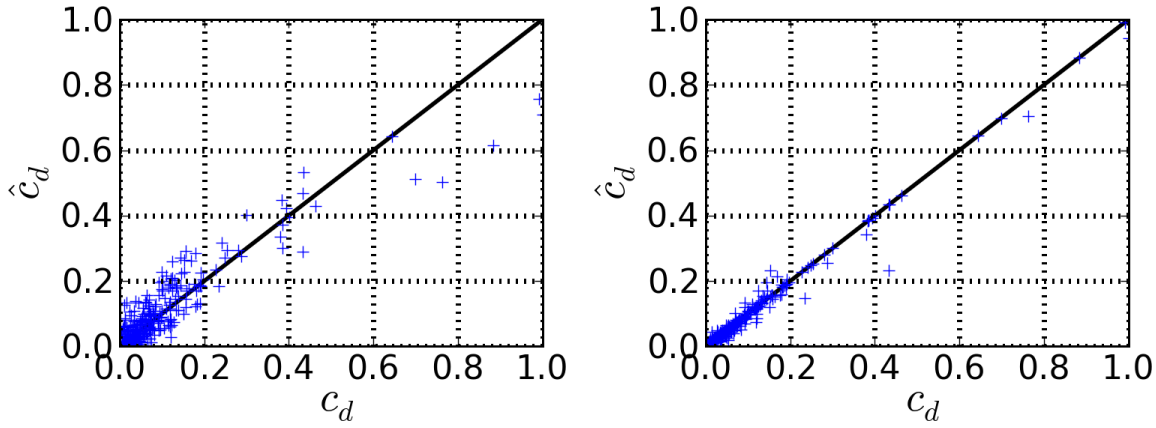


(a) RS approximation on experimental data (b) SVR approximation on experimental data

Figure 6.6: Comparison of c_l metamodels on experimental data

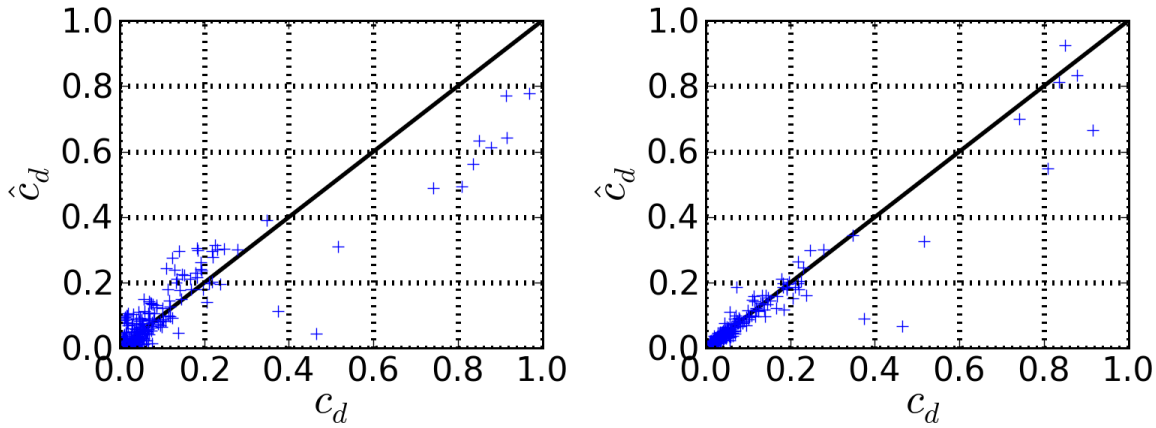
6.1.6 Drag comparison

In this section, the c_d approximations are compared to the simulated results obtained with XFOIL. Figure 6.7 shows the results obtained at the OD design points when evaluated with the RS and SVR metamodels in comparison to XFOIL, while Figure 6.8 shows the drag approximations vs. XFOIL on the ED design points for both the RS and SVR metamodels. The SVR error on the original and experimental data is quite similar, which corresponds to the AAE error for the drag in section 6.1.3. The RS errors on the OD and ED are also similar, which is expected when enough design points are used for an RS model. However, the SVR drag approximation is significantly more accurate than the RS approximation. It is clear that the RS drag approximation does not follow a second order trend.



(a) RS approximation on original data (b) SVR approximation on original data

Figure 6.7: Comparison of c_d metamodels on original data



(a) RS approximation on experimental data (b) SVR approximation on experimental data

Figure 6.8: Comparison of c_d metamodels on experimental data

6.2 Optimizing the C_p values

Once the metamodels are available, it is possible to optimize the wind-turbine for C_p. The optimal C_p value is found with the gradient-based optimizers by utilizing the BEM theory in conjunction with the RS and SVR metamodels for c_l and c_d across the 4-digit NACA range. The C_p function was found to have many local optima, and as a result the optimal C_p seldom converged to a common design point when starting from different initial design points. These optima can be caused by noise in the C_p function, a restriction in the design constraints or local maxima in the design space.

To start with, a total of 30 initial design points describing the complete 4-station blade was created with a Design Of Experiments (DOE). The DOE utilized an Optimum Latin

Hypercube (OLH) design because of its space filling characteristics. The purpose of using more than one initial point is to determine whether the final C_p function has local optima, and where a global optimum might exist. The 30 initial design points were optimized with the Response Surface (RS) approximation and the Support Vector Regression (SVR) approximation.

Each initial point was optimized with the three gradient-based optimization methods SLP, SQP and MMFD. However, the results of these optimizations were very poor. One reason for the poor results is that with infeasible initial blade design points the BEM program would crash as a result of diverging induction factors caused by α's beyond of the limits of the metamodel boundaries. Another reason for poor results is caused by numerical noise in the C_p function. The results of these initial points were then used as an exercise to find feasible boundaries on the design variables of the blade.

With the knowledge gained from this exercise, a 2^{nd} set of data was created by changing the boundaries on the θ and c values, as shown in Table 6.2. Here the subscripts u and l denote the upper and lower boundaries on the design variables. The bounds on the k, l and mm values remain the same (Table 5.5). With improved boundaries on the initial blade design points, a DOE was done with the OLH method again to create second set of 30 initial design points. With these improved initial blade design boundaries an improved performance could be obtained from the optimizer.

Table 6.2: Boundaries on 2^{nd} set of initial design points

	Station 1	Station 2	Station 3	Station 4
θ_l	10	5	1	0
θ_u	25	10	5	1
c_l	0.7	0.6	0.3	0.2
c_u	0.8	0.75	0.6	0.4

6.3 Using SVR to optimize for C_p

Once a sufficiently accurate SVR approximation and initial points are available, the C_p value can be optimized. The optimum C_p values is found by coupling the BEM theory and the SVR approximations to the gradient-based optimizers. The results from these three gradient-based optimization techniques discussed in sections 2.1.4,2.1.5 and 2.1.6 are presented in this section. As a result of many optima being found, many different

blade designs are available for use. However, only one suitable result is discussed in more detail in this thesis.

6.3.1 Resultant C_p values

Each of the initial blade designs were optimized with the three gradient-based optimizers, namely SLP, SQP and MMFD. The final results of the optimization study are summarized in Fig. 6.9. In this figure the optimized C_p values for each of the 30 design points is shown for each of the 3 gradient-based techniques. The figure shows the resultant C_p values ranging from smallest to largest for each optimization technique. A number of points again did not yield good results as a result of numerical noise in the C_p function and the effect of poor initial points causing the induction factors to diverge.

The MMFD technique is the least affected by numerical noise, and yields the best and most consistent overall results. Many of the optimized C_p values obtained with all three techniques are ≈ 0.525. However, the design variables at these optimized C_p values were all found to have completely different blade designs. The different optimal design points mean that the C_p function has a large number of local optima, and that many different airfoil shapes yield similar C_p's. An advantage of different optimal design points means that the designer has a large number of different blade profiles to choose from.

The times required for each design point by each gradient-based method varied vastly. In Table 6.3 the time ranges are shown that are typically required for the optimization of one design point by the respective gradient-based method. The calculations are done on a 2.2 GHz 64 bit Dual Core AMD processor.

Table 6.3: Time to optimize for C_p with SVR metamodels

Method	Time ranges
MMFD	900s - 7500s
SLP	500s - 8500s
SQP	600s - 2500s

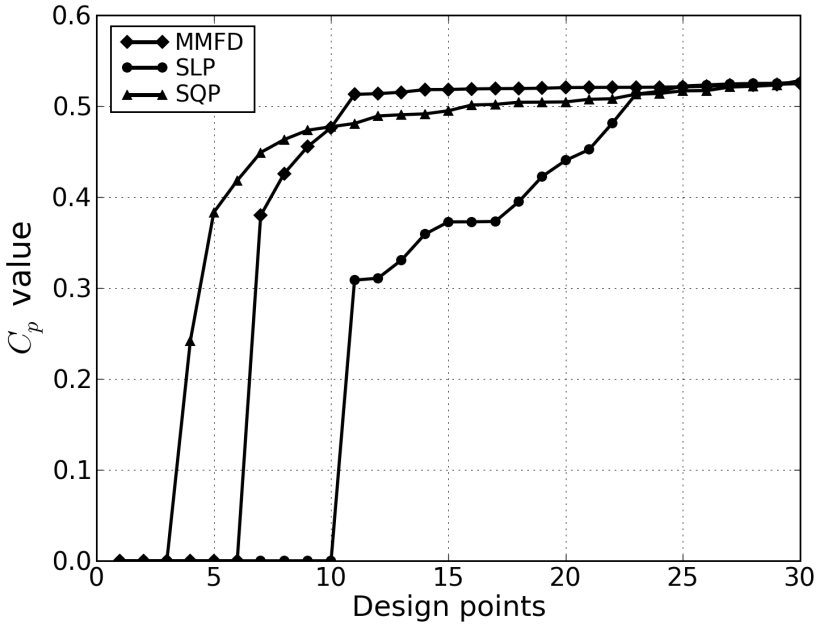


Figure 6.9: Final C_p values obtained

6.3.2 Blade optimization with SVR

In this section only one of the optimized blades are discussed. Each blade is described by the NACA coefficients k, l, mm, the angle of attack (α) and the chord lengths (c) for all 4 stations, amounting to a total of 20 design variables describing the blade. The optimized design point that is discussed has initial design variables as shown Table 6.4, and has an initial $C_p = 0.49$, which is already a feasible design. Many of the initial C_p values of the 30 initial points vary around 48%, as a result of being chosen in regions specifically found to yield good results, as discussed in Section 6.2.

Table 6.4: Initial design points

	Station 1	Station 2	Station 3	Station 4
k	1.24	5.58	4.34	6.21
l	1.86	0.93	1.24	6.21
mm	11.79	3.	11.21	8.86
θ	21.9	8.10	3.62	0.93
c	0.74	0.74	0.40	0.23

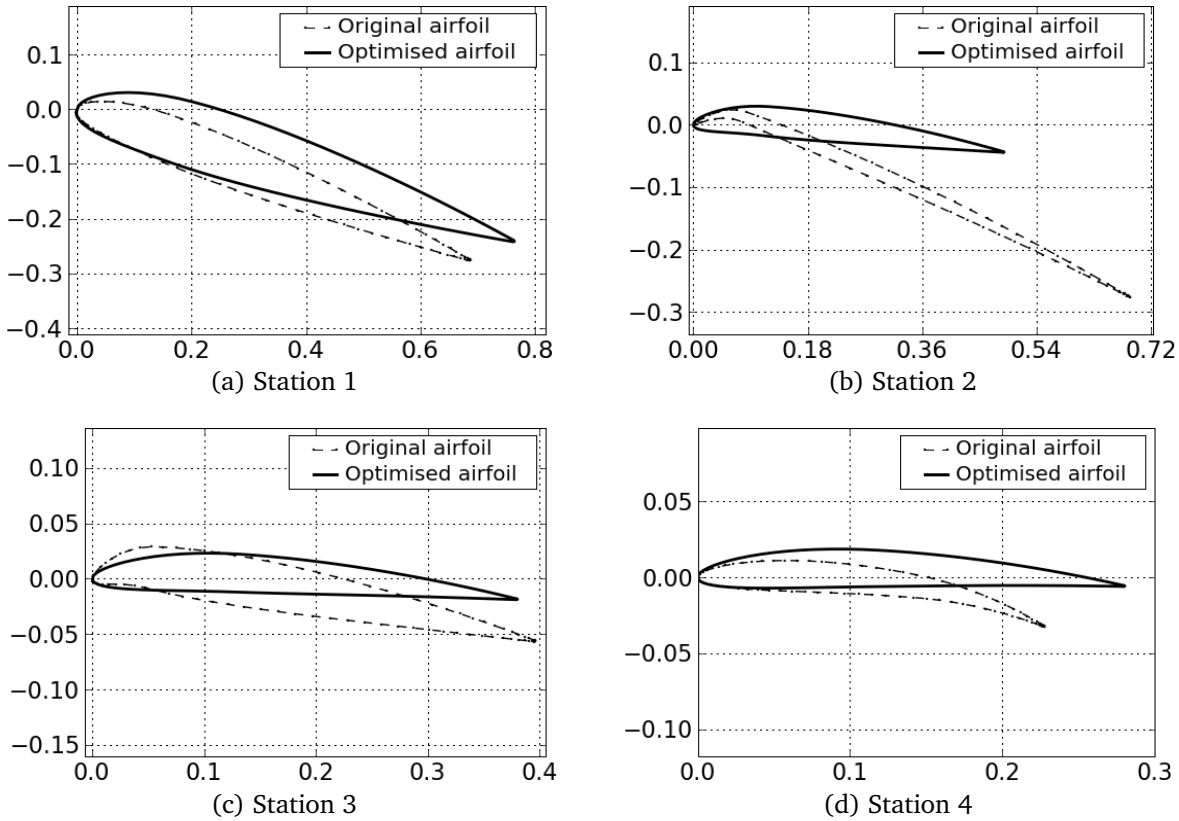
The final design point used is found using the Sequential Quadratic Program (SQP) optimizer. Many of the other optimized blades with $C_p \approx 0.525$ with any of the optimization techniques can be used just as well. The optimized design variables for the

blade are shown in Table 6.5. In this table, each design variable has been changed from its original value. The optimal design variables have a decreasing flow angle θ, as well as a decreasing chord length c. These are favourable characteristics present in traditional wind-turbine implementations found in literature [6]. One other favourable characteristic present in this design is the decreasing thickness factor, mm, which means that the structure is more structurally feasible than it would be with an increasing thickness. These characteristics were not specifically implemented in the optimization routine, but can be easily implemented by the designer.

Table 6.5: Optimized design variables

	Station 1	Station 2	Station 3	Station 4
k	1.88	3.2	3.01	2.94
l	2.22	2.29	3.45	4.13
mm	14.93	9.8	9.09	8.96
θ	17.54	5.15	2.8	1.18
c	0.8	0.49	0.38	0.28

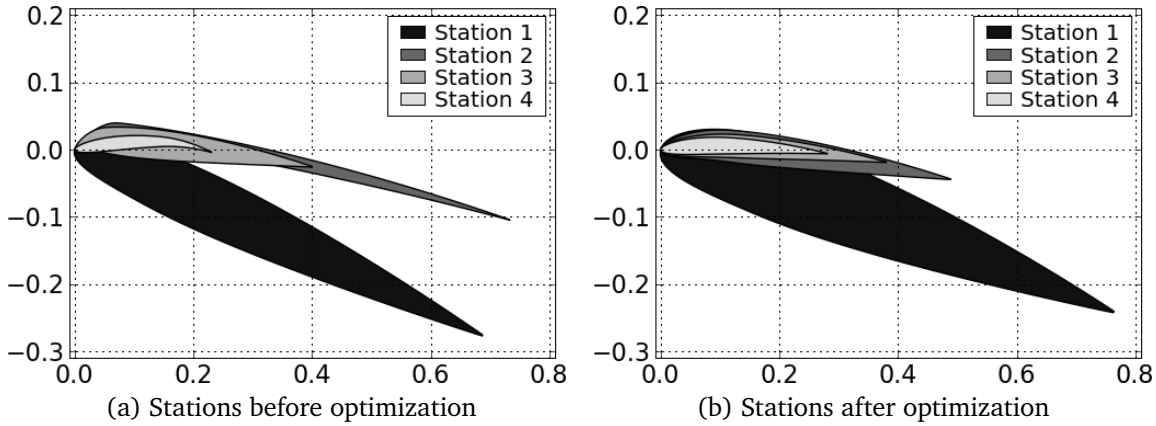
The optimized blade is graphically represented in terms of its four radial stations in Fig. 6.10, where the initial and final airfoils are plotted against each other. Each station shown in the figure shows the optimized NACA profile, angle of attack and chord length. In Fig.'s 6.11a and 6.11b the four stations are plotted against each other showing the relative shapes before and after optimization. The results represented here are not subject to any relative constraints, i.e. no radial station has constraints relative to other radial station. Any relative constraints can simply be implemented in the optimization program. However, no such constraints were implemented as the principle purpose is to demonstrate the use of SVR as a metamodelling technology.



(a) Station 1 (b) Station 2

(c) Station 3 (d) Station 4

Figure 6.10: Airfoil optimization



(a) Stations before optimization (b) Stations after optimization

Figure 6.11: Relative sizes of airfoil shapes at individual stations

6.3.3 C_p verification over design wind speed

Once a suitable C_p and corresponding design point have been found with the SVR meta-models for c_l and c_d, it is necessary to verify the accuracy of the resultant C_p for the wind turbine with XFOIL. Still using 11 wind-speed stations, the average C_p is evaluated with the BEM program over $n = 11$ stations using all three techniques, namely RS, SVR and XFOIL. Table 6.6 shows the results obtained from the two approximation methods and the results obtained from XFOIL.

Table 6.6: Resultant C_p values

Evaluation method	$\overline{C_p} = \frac{\sum C_p}{n}$	Error	Time
XFOIL	0.514		$\approx 1200s$
SVR approximation	**0.525**	2.1%	$\approx 10s$
RS approximation	0.536	4.3%	$\approx 0.5s$

The percentage error is determined from as follows:

$$\text{Error} = \left| \frac{\overline{C_{p\,XFOIL}} - \overline{C_{p\,SVR/RS}}}{\overline{C_{p\,XFOIL}}} \right| \tag{6.1}$$

The time values given in Table 6.6 are for a single evaluation of C_p for the corresponding method with the initial induction factors set to 0. It is clear from these time values that using XFOIL to evaluate c_l and c_d values in the BEM program is infeasible, as roughly 100 C_p evaluations are required in a typical optimization routine.

The C_p characteristics can be seen in Fig. 6.12, where the C_p is plotted against the wind-speed over the specified axial wind velocity range. In the figure, the same design point is evaluated for C_p with the RS and SVR approximation and compared to the C_p obtained with XFOIL when used in conjunction with the BEM theory. The C_p value obtained from XFOIL and the SVR approximation are seen to peak within this range, unlike the RS approximation.

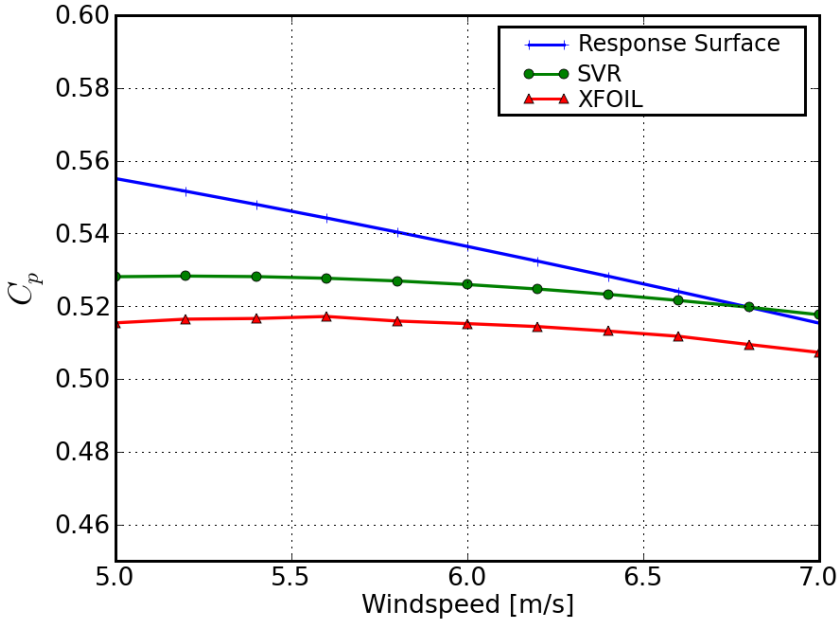


Figure 6.12: Final C_p obtained with SVR

6.3.4 Contribution of each station to C_p

Each radial station contributes a part to the final C_p. Figure 6.13 shows the contribution of the four stations used in the approximation. The two central stations contribute the majority of the C_p. Although the two outer stations contribute less, the root station is typically used to create more power at start-up, and the tip station can be used to induce stall at high wind speeds [6].

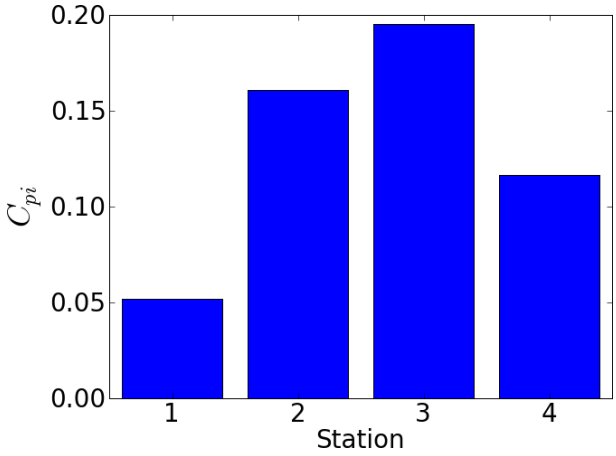


Figure 6.13: Contribution of stations to final C_p

6.3.5 Reynolds number and α at final C_p

At the final design point, it is useful to know at what angles of attack and at what Reynolds numbers the stations are operating at. In Fig. 6.14 the angles of attack for the four stations are shown across the wind-speed divisions. The angles of attack are all smaller than 10°.

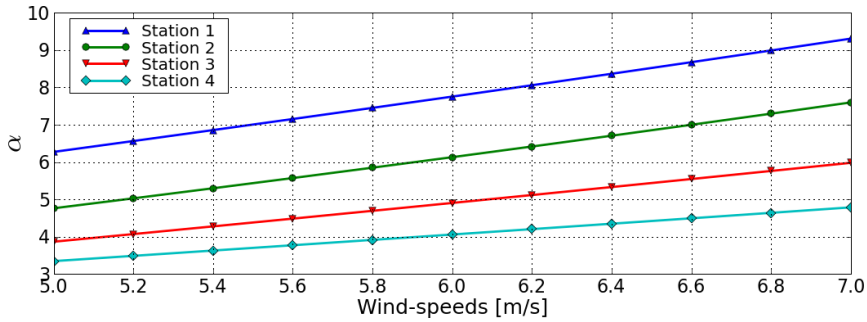


Figure 6.14: Angles of attack in degrees

The Reynolds number for the 4 station across the design wind speed is shown in Fig. 6.15. The Reynolds number increases with an increase in the corresponding α values. It is interesting to note that station 4 has a lower Reynolds number than station 3, even though it is operating at a higher velocity. The lower Re can be attributed to a smaller chord length.

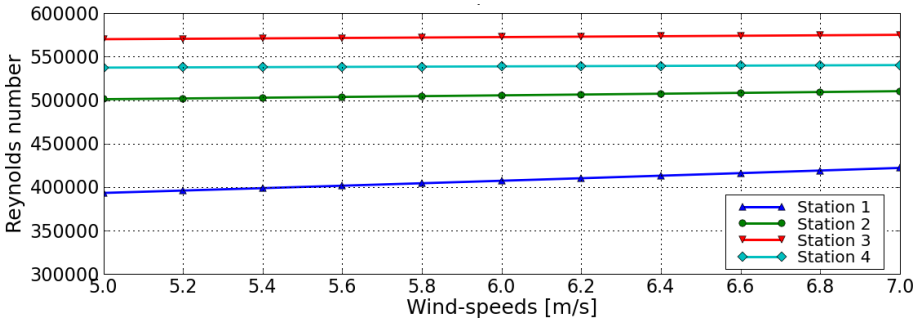


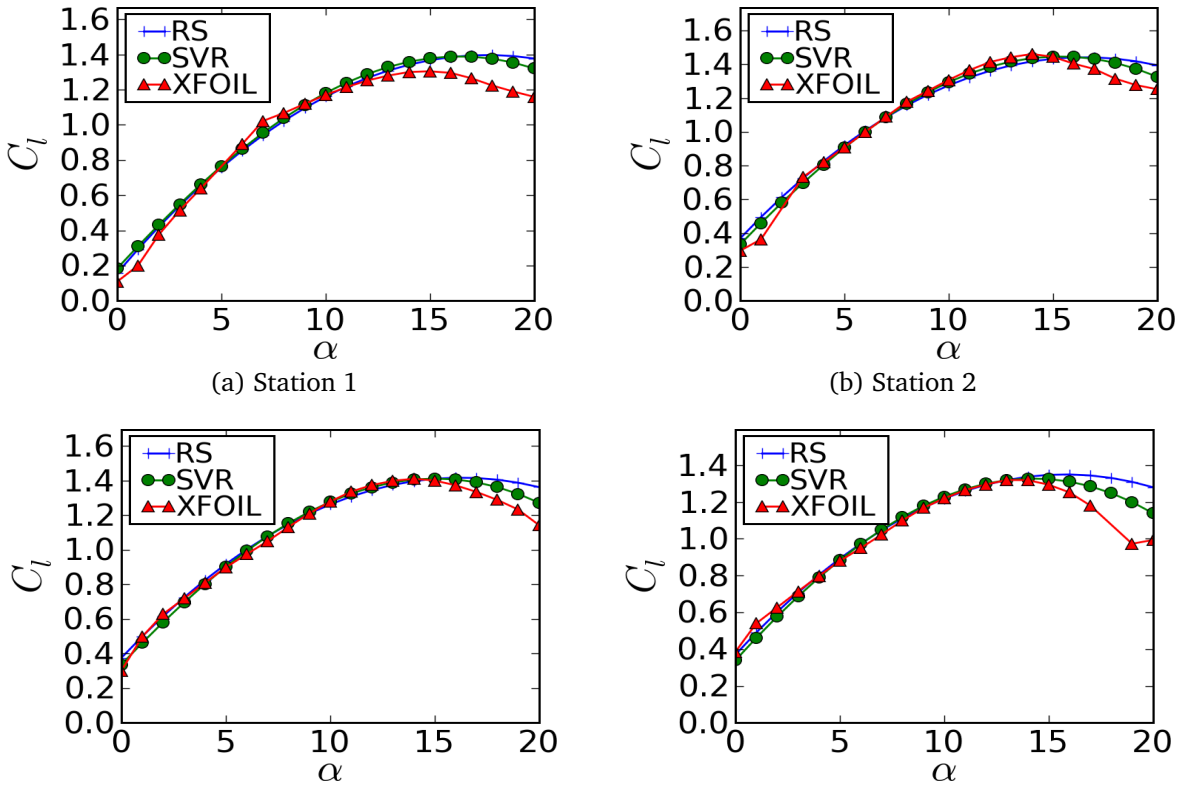
Figure 6.15: Reynolds number across design wind-speed

6.3.6 Comparing c_l at optimal design point

Once the optimal design point and the optimal C_p are available, the c_l and c_d values are evaluated with RS and SVR and are compared to the results obtained with XFOIL across the range of angles of attack from 0 to 20. The comparison is done with the optimal

NACA profiles and optimal chord length, and is similar to taking four snapshots through the design space to determine the accuracy of the SVR and RS metamodels.

Figure 6.16 plots the c_l approximations against angles of attack ranging from 0 to 20 degrees. The c_l function has accurate approximations by both the RS and SVR metamodels. At high angles of attack, a discrepancy starts occurring between the metamodels and XFOIL. The discrepancy can be attributed to fact that the majority of XFOIL evaluations for the training points failed at high angles of attack. The c_l has a quadratic nature, and does not seem to justify the use of a more complex metamodelling technique like SVR.



(a) Station 1

(b) Station 2

(c) Station 3

(d) Station 4

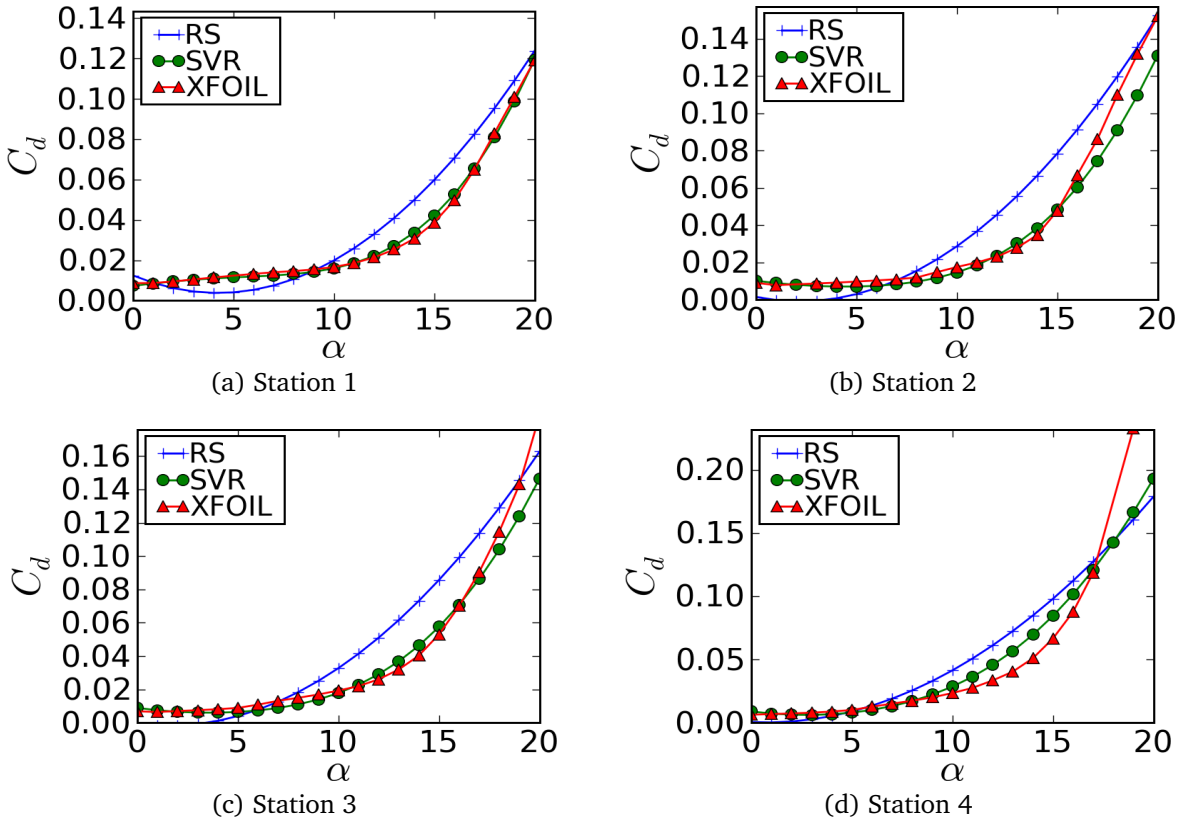
Figure 6.16: Comparison of the c_l approximation with XFOIL

6.3.7 Comparing c_d at the optimal design point

The metamodels of the drag coefficient is verified with XFOIL at the chosen optimal design point in the same way as the lift coefficient. The drag approximations vs. the XFOIL evaluations across the range of angles of attack from 0 to 20 are shown in Fig. 6.17. Again, at high angles of attack the accuracy of both metamodels starts deteriorating. At low angles of attack, the SVR approximation closely approximates the values

obtained with XFOIL. The α values used in the chosen optimal design are all below 10 degrees as shown in section 6.3.5, thus the inaccuracies developed at high α values do not significantly affect the final C_p.

In contrast to the SVR metamodel, the RS metamodel is inaccurate over the entire range of α values. The c_d function can therefore not be described accurately by a 2^{nd} order approximation. A metamodelling technique better suited to non-linear approximations is therefore a more appropriate option in approximating the drag function. This corresponds with the research conducted by Clarke [5], in which Response Surface approximations were found to be more accurate for linear approximations and where SVR approximations were found to be more accurate for non-linear problems.



(a) Station 1

(b) Station 2

(c) Station 3

(d) Station 4

Figure 6.17: Comparison of the c_d approximations with XFOIL

6.4 Using RS to optimize for C_p

The alternative to optimizing the C_p with SVR is to optimize for C_p with the RS approximation. With the same initial point given in Table 6.4, a completely different final

design point is reached with RS when using the SQP optimizer. Below in Table 6.7 the final design point is given.

The final design point lies in a feasible area, and appears to be a feasible design, with the average C_p over the design wind-speed is 0.587. However, when the design point obtained for this approximation is compared to the result obtained with XFOIL, the difference is significant, as shown in Fig. 6.18. The error here is larger than at the final DP obtained with SVR. This can be attributed to the optimizer exploiting the areas in which the RS approximation is poor.

Table 6.7: Final design point with RS approximation

	Station 1	Station 2	Station 3	Station 4
k	5.18	1.79	1.6	0.78
l	4.46	4.16	4.16	3.96
mm	12.01	11.17	11.37	11.13
θ	17.54	5.15	2.8	1.18
c	0.8	0.8	0.675	0.58

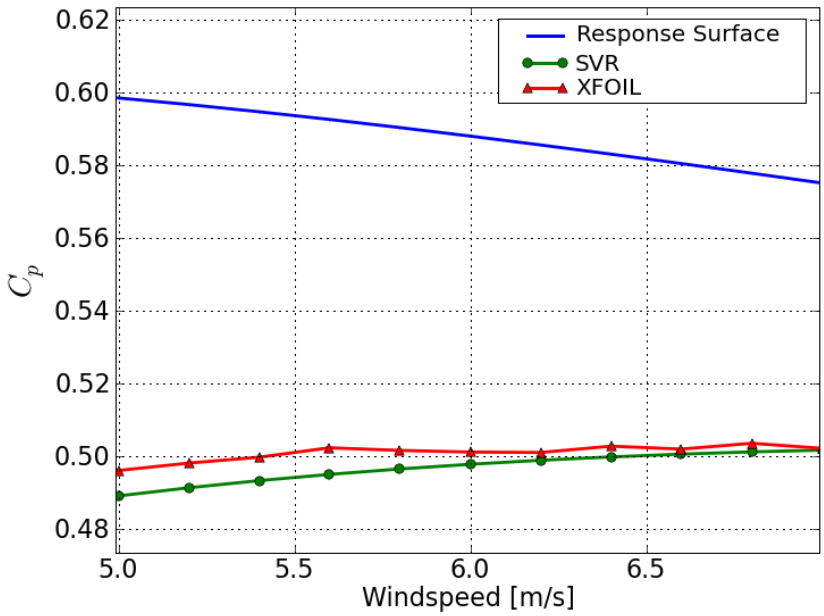


Figure 6.18: Final C_p with RS

Using Eq. 6.1 to determine the error at this design point using RS yields an error of $\approx$ 17%, while the SVR metamodel yields an error of $\approx$ 0.7%. As a result of the large discrepancy at this design point, the result will not be explored further. However, it is

apparent that the 2^{nd} order RS metamodel is not a feasible option for optimizing for C_p in this context.

6.5 Conclusion

In this section, the creation of lift and drag metamodels for the NACA 4-digit series with SVR and RS technology was discussed. It is found that the lift is modelled well with both technologies, while the drag is poorly approximated with a 2^{nd} order RS approximation. The metamodels are used in conjunction with the BEM theory to optimize a wind turbine. A total of 30 initial design points were optimized with the three gradient-based optimizers. Many different maxima were encountered with similar C_p values, due to local maxima and numerical noise. The advantage of the many local maxima with similar C_p values is the designer can select from different designs to obtain the same performance. For demonstration, one optimum was chosen and compared to the C_p value found with the RS metamodel and XFOIL. The error in the C_p when using the SVR and RS metamodels was found to be 2.1% and 4.3% respectively.

The RS metamodel was also used to optimize the given 30 initial design points for C_p. The majority of the optimized design variables obtained C_p values in the order of 58%. These values acquired with the RS metamodel are unrealistically high as a result of the optimizer exploiting the poor RS c_d approximation. Each of the optimized design points that was found when using the RS metamodel, despite having similar C_p values, also have completely different optimized design points.

Chapter 7

Conclusion

7.1 SVR as a metamodelling technology

THIS thesis explores the use of Support Vector Regression (SVR) as a metamodelling technology. It was found that an accurate SVR approximation is subject to having a sufficiently high C value. For non-linear approximations, a Gaussian Radial Basis kernel was utilized, and it was found that the best approximation is subject to selecting the correct σ value. Furthermore, it is possible to improve the error by using enough training points.

7.2 Lift and drag metamodels

Lift and drag metamodels were created with SVR and RS for the NACA 4-digit series within specified design limits. A total of 314 points were used to create the metamodels. After scaling the c_l training points and finding the optimal parameters for the metamodels, the best Average Approximate Error (AAE) with SVR was found to be 0.023, while the best AAE with RS was found to be 0.027. The lift function can therefore be accurately approximated with a 2^{nd} order polynomial, and does not require the use of a more advanced metamodelling technology like SVR. Following the same methodology, SVR and RS metamodels were created for the c_d function. The best AAE of the SVR metamodel was found to be 0.018, while the RS metamodel yielded an AAE of 0.047. It was found that SVR models the drag response sufficiently accurately to use in conjunction with the optimizer, while the drag function cannot be modelled accurately with a 2^{nd} order polynomial function.

In conclusion, both the RS and the SVR approximations can model the c_l function accurately, while only the SVR approximation can model the c_d function with sufficient accuracy to use in the optimization procedure.

7.3 BEM theory

The BEM algorithm was implemented to evaluate the power coefficient, C_p, of a wind-turbine blade that uses a variety of NACA profiles at specified environmental conditions. The wind-turbine is described by a total of 20 design variables which are provided to the program implementing the BEM theory. The BEM algorithm was implemented in Python with three minor modifications to the one specified in Hansen [3]. Firstly, convergence is specified to occur when the value of C_p converges. Secondly, the number of iterations required in a single optimization routine was reduced by storing the converged values from one converged BEM routine call and using the stored induction factors for a subsequent BEM call to determine the C_p. The motivation for storing induction factors is that it substantially reduces the time to convergence. Lastly, the c_l and c_d values are determined from the metamodels instead of a chart or by calling a simulation program like XFOIL.

Even though XFOIL is already a fast method of approximating the lift and drag coefficients for airfoils, using the metamodels instead of using XFOIL meant a great reduction in computational cost. Finding a single C_p with XFOIL requires approximately 1200 seconds, while the SVR and RS metamodels require approximately 10 and 0.5 seconds respectively. It is therefore infeasible to use XFOIL to optimize a wind-turbine blade in this manner, as the optimization routine requires in the order of 100 C_p evaluations to find an optimum.

The accuracy of the C_p value was found to be subject to a number of factors. Firstly, the greater the number of iterations in the BEM program, the greater the accuracy of the C_p value, which means a longer time to converge. Another factor influencing the C_p accuracy is the number of wind-speed divisions. The greater the number of wind-speed stations, the greater the C_p accuracy. However, more wind-speed divisions mean longer evaluation times for each C_p. In the final optimization procedures, 11 wind-speed stations were utilized and a C_p convergence accuracy of 10^{-5}.

To further improve the two-dimensional metamodels derived from XFOIL, metamodels can be derived from more accurate CFD software using the same method described in this thesis.

7.4 Finding the optimal C_p

To demonstrate the use of SVR in a real-life application, the BEM program using the SVR metamodels was coupled to the commercial gradient based optimizers implemented in VisualDOT. The 3 gradient based techniques, SLP, SQP and MMFD were used in conjunction with the BEM program to determine the optimal C_p values. The optimization procedure was found to be affected by numerical noise, local maxima as well as diverging induction factors due to infeasible design points. All three techniques operated between the same time limits required to complete an optimization problem, with MMFD being least affected by the numerical noise. The presence of numerical noise and diverging induction factors meant that many final blade designs are not optimal designs. A large number of local optima were found to exist in the C_p function, with many different final blade designs having similarly high C_p values. The large number of different blade designs with similar C_p values means that the designer has many designs from which to choose.

The optimum mean C_p value is found at $C_p = 0.525$. This is 52.5% of the total energy in the wind, and is roughly 7% less than the Betz limit [3]. It seems as though the NACA 4-digit series is limited from getting much higher than the optimized value of $C_p = 0.525$. However, if an effective method can be found to eliminate the numerical noise, perhaps a better blade performance can be obtained. The noise in the C_p function is caused by the induction factors that need to evaluated iteratively until they converge. One method of eliminating the numerical noise would be to create a metamodel of the C_p function.

The final design point that was chosen was found using the SVR metamodels. An initial design point was optimized with the SVR metamodel in conjunction with the BEM program and the optimizers, which yielded an optimum. The optimum design point chosen was evaluated with SVR and XFOIL, which yielded a 2.1% error. At the same design point, the C_p found by using the RS metamodel yielded a 4.1% error. However, when the RS metamodel was used in conjunction with the BEM program to optimize the same initial design point, the optimal design point has an error of $\approx$ 17%. Using the SVR metamodel to evaluate the optimal design point found with the RS metamodels yielded an error of 0.7%, which is significantly smaller than the error obtained with the RS metamodel. The large error means that the RS metamodel was an infeasible option to use for optimizing the blade.

7.5 Meeting original objectives

The two objectives were to explore the use of SVR as a metamodelling technology and to utilize it in a real-life optimization problem. It was found that SVR can be used to model highly non-linear responses, as well as problems with higher dimensionality as in the chosen demonstration problem which had 5 dimensions for each metamodel.

The application chosen was to optimize a wind-turbine using the 4-digit NACA series for airfoils. The lift and drag coefficients were modelled with SVR and RS, and used in conjunction with the BEM theory to optimize the power coefficient of the turbine. In utilizing SVR in the wind-turbine application, it was found that the optimization took a fraction of the time to evaluate a C_p value with the SVR metamodel instead of with XFOIL. Although a small error existed between the SVR and XFOIL approximation, the advantage of using SVR as a metamodelling technology in terms of computational expense is apparent.

7.6 Future work

Thus far, only a theoretical model for the optimal blade is available. Potential future work will entail the creation of a CFD model to verify the results obtained with the BEM theory. Future topics that can also be explored include:

- Finding the most efficient algorithm to implement the SVR problem between all the ones that currently exist. This includes the promising SMO algorithm described by Nello [15].
- Exploration and comparison of other kernel types currently used. More suitable kernel types can also be created.
- The creation of a metamodel of the C_p function through the BEM theory, instead of approximating the lift and drag functions. This will require a well designed DOE procedure to create the training points that describe the entire blade.
- The creation of a metamodel of the other NACA airfoil ranges, for example the 5-digit series through the 8-digit series.
- The creation of a metamodel of unique airfoils in the manner that the airfoils were optimized by Cencelli [6].

Bibliography

[1] Vanderplaats, G.N.: *Numerical Optimization Techniques for Engineering Design*. 4th edn. Vanderplaats Research and Development, Inc, 2005.

[2] Smola, A.J. and Schölkopf, B.: A tutorial on support vector regression. *Statistics and Computing*, vol. 14, pp. 199 – 222, 2005.

[3] Hansen, M.O.L.: *Aerodynamics of Wind Turbines*. James&James (Science Publishers) Ltd, 2000.

[4] Fang, H., Rais-Rohani, M. and Horstemeyer, M.F.: Multiobjective crashworthiness optimization with radial basis functions. *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, August 2004.

[5] Clarke, S. M. Griebsch, J.H. and Simpson, T.W.: Analysis of Support Vector Regression for Approximation of Complex Engineering Analysis. *ASME Journal of Mechanical Design*, vol. Vol. 127, No. 6, pp. 1077 - 1087, 2005.

[6] Cencelli, N.A.: *Aerodynamic optimisation of a small-scale wind turbine blade for low windspeed conditions*. Master's thesis, Stellenbosch University, 2006.

[7] Drela, M.: XFOIL 6.9, MIT Aero & Astro, Harold Youngren, Aerocraft, Inc. 30 Nov 2001.

[8] Bates, S.J., Sienz, J. and Toropov, V.V.: Formulation of the optimal latin hypercube design of experiments using a permutation genetic algorithm. *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2004.

[9] Viana, F.A.C., Balabanov, V., Venter, G., Garcelon, J. and Steffen, V.: Generating optimal latin hypercube designs in real time. In: 7^{th} *World Congress on Structural and Multidisciplinary Optimization*.

[10] McDonald, D.B., Grantham, W.J. and Tabor, W.L.: Response surface development for global/local optimization using radial basis functions. *AIAA 2000-4776*, 2005.

[11] Martin, J.D. and Simpson, T.W.: Use of Kriging Models to Approximate Deterministic Computer Models. *AIAA JOURNAL*, vol. 43, No. 4, 2005.

[12] Myers, H.M. and Montgomery, D.C.: *Response Surface Methodology, Process and Product Optimization Using Designed Experiments*. 4th edn. John Wiley & Sons, Inc, 1995.

[13] Vanderbei, R.J.: LOQO: An interior point code for quadratic programming. 1998.

[14] Schölkopf, B. and Smola, A.J.: *Learning with Kernels*. The MIT Press, Cambridge, Massachusetts, London, England, 2002.

[15] Cristianini, N. and Shawe-Taylor, J.: *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2003.

[16] Hsu, C.W., Chang, C.C. and Lin, C.J.: A Practical Guide to Support Vector Classification. 2007.

[17] Gunn, S.R.: Support Vector Regression for Classification and Regression. 1998.

[18] Jacobs, E. N., Pinkerton, R. M. and Ward, K. E.: The characteristics of 78 related airfoil sections from tests in the variable density wind tunnel. Tech. Rep., NACA, 1932.

[19] Spera, D.A.: *Wind Turbine Technology*. ASME Press, 1995.

[20] van Rossum, G.: *Python Tutorial*. 2006.