Quality assessment of data-based metamodels for multi-objective aeronautic design optimisation

Timur Topuz

November 2007

vrije Universiteit
Faculteit der Exacte Wetenschappen
Studierichting Bedrijfswiskunde en Informatica
De Boelelaan 1081a

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Design of complex products, such as aeronautic systems and components, requires extensive analyses of the product's behaviour. These analyses can be done by physical experiments, i.e. wind tunnels, or by computer experiments, i.e. analyses based on computational model simulations. In the past decades, engineering analysis has relied to an increasing extent on complex computer models and simulation codes, such as finite element and computational fluid dynamic analyses, to simulate the performance of the product under consideration [42]. These computer experiments can be very accurate in their simulation of physical experiments, but in that case go to great expense in computational costs (time and money).

Design analysis usually takes into account multiple evaluations of a design in order to find the best performance of the product, which results in multiplication of the computational costs of the computer experiments. Therefore, other computationally cheaper methods for efficient representation of the product behaviour are desired. For this purpose approximation methods can be used to create so-called metamodels: simplified models that provide computationally efficient representations of the original design analysis. These metamodels are intended to efficiently and accurately predict the characteristics of a product, based on the properties of a design.

Metamodels are created by combining two pieces of information:

1. information about the product behaviour, expressed by a dataset, and

2. interpolation information, expressed by analytical mathematical functions.

The product behaviour information can be evaluated in traditional ways (eg. physical or computer experiments), the results of which yield datasets of the product behaviour.

The interpolation information can be of many different forms. A commonly used type of metamodel is the polynomial model, which is based on a polynomial approximation function in combination with a least-squares error regression technique [33]. Many other types of metamodels are also available. The types of metamodels considered in this study are:

- Polynomial models
- Kriging models
- Radial Basis Functions
- Multivariate Adaptive Regression Splines
- Support Vector Regression
- Neural networks

Both the quality and quantity of the dataset as well as the choice for a specific type of a metamodel have an influence on the quality of the approximation, i.e. how well the metamodel represents the design analysis.

The quality of the metamodel, i.e. the accuracy of the prediction of the metamodel, can be assessed by applying the metamodel to an additional dataset of product behaviour, also called a validation set. With this assessment it is determined how well the metamodel predicts unknown product behaviour, i.e. the product behaviour covered by the validation set and *not covered* by the dataset with which the metamodel was created. The quality of the metamodel depends on the appropriateness of the type of metamodel for a certain dataset. Jin et al. [15] provide a good overview of which type of metamodel to use for which dataset.

However, if an additional dataset of product behaviour is available, it is desirable to incorporate these data points into the process of creating the metamodel. Since more information will yield a more accurate metamodel. This way a real validation set is never really available.

Due to the characteristics of some of the metamodels, assessing the quality on known data points is also impossible. This is because some metamodels can predict known data points exactly. Therefore, an alternative method for assessing the quality of a metamodel is to split up the available data points. This way of quality assessment of metamodels is also called 'cross-validation' [31].

This report presents an investigation of metamodels known in the literature as described in Chapter 3. First, in Chapter 2 the context in which the metamodels will be studied is outlined. In Chapter 4 the methods for validating metamodels are described, where special attention will be given to cross-validation techniques.

Another part of this study is the implementation of some selected metamodels in a software tool. The implementation and usage of this software tool are described in Chapter 5. In Chapter 6 it will be made clear which settings are chosen in the implementation of the software. Verification of the software by several test problems are done in Chapter 7.

The methods developed and described are also applied to an example aeronautic design problem as described in Chapter 8. Chapter 9 gives the general conclusion.

Chapter 2

Multi-objective optimisation

2.1 Design Optimisation

A product is defined by its properties. These properties affect the functionality of a product. In product design the effects of the product properties at its functionality are evaluated by physical or computer experiments. And so, these experiments indicate the effectiveness of the variation of the product properties, and lead to a satisfying product design.

In aeronautic design, many different physical and computer experiments are in use. Computer experiments, for example based on finite element methods (FEM) [27] or computational fluid dynamics (CFD) [1], make it possible to evaluate a design without actually having to build the product. These computer experiments are very accurate in the simulation of the physical experiment, but may go in great expense in computational costs. In product design it is usually desired to evaluate multiple designs to be able to compare different designs. Computer experiments are not always efficient for this multiple design analysis, because of their time-consuming character. Therefore, more efficient models are needed that approximate the design analysis.

The goal in product design is usually to optimize the functionality of the product by adjusting its properties. For this purpose, the dependency of the functionality or performance of the product on its properties is expressed as a mathematical model. For example in aircraft wing design, the dependency of the weight and drag of the wing on its span (length) and sweep (angle) can be expressed by mathematical models based on structural mechanics and aerodynamic theories.

As illustrated with this basic wing design example, see figure 2.1, the optimisation of the functionality of a product generally involves more than one characteristic, e.g. for optimum wing functionality, both the wing weight

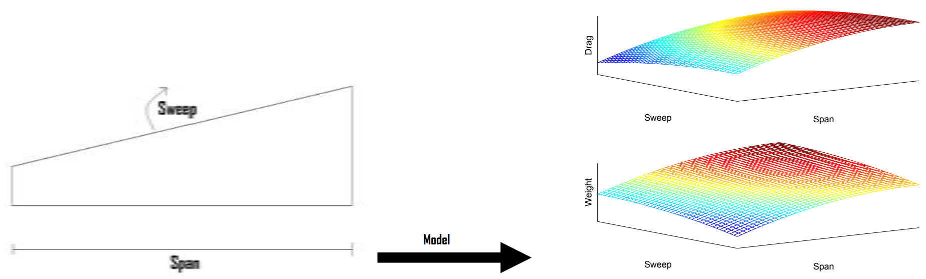


Figure 2.1: Multi-objective wing example

and its drag should be minimized. Simultaneously optimizing multiple, and often contradicting, characteristics requires specific optimisation algorithms, and in general many design evaluations are needed. Such simultaneous optimisation is called multi-objective optimisation (MOO).

2.2 Pareto Optimality

In multi-objective optimisation problems, the decrease of one objective often causes an increase of another objective. Generally, one single global optimum, which has an optimum value for all objectives does not exist in multi-objective optimisation. Instead a collection of optima exists, which represents the set of optimal points. This is called a Pareto optimal set [35].

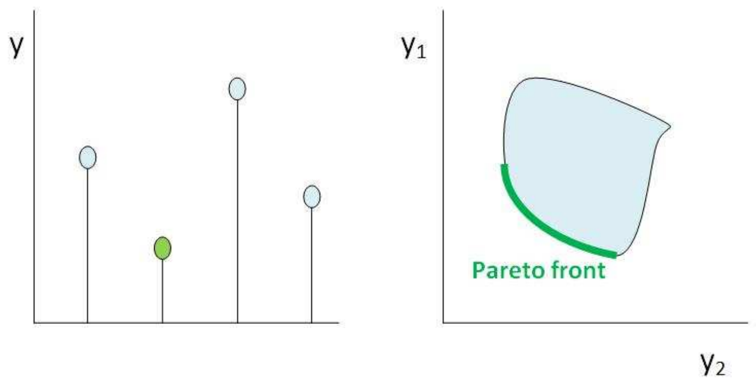


Figure 2.2: Optimal points in single-objective optimisation (left) and multi-objective optimisation (right)

As can be seen in figure 2.2, in contrast with single-objective optimisation, where the optimum is simply represented by the minimum (or maximum) value of the objective function, for multi-objective optimisation a different definition of optimality is needed, one that respects the integrity of all the separate objectives. The concepts of Pareto optimality helps defining such optimal solutions in a rational way.

Considering a minimization problem, a point $\boldsymbol{x}^*$ is Pareto optimal if there does not exist another point $\boldsymbol{x}$ such that $f_i(\boldsymbol{x}) \leq f(\boldsymbol{x}^*)$ for all $i = 1, \ldots, k$, where k is the number of objectives, and $f_j(\boldsymbol{x}) < f(\boldsymbol{x}^*)$ for at least one $j = 1, \ldots, k$.

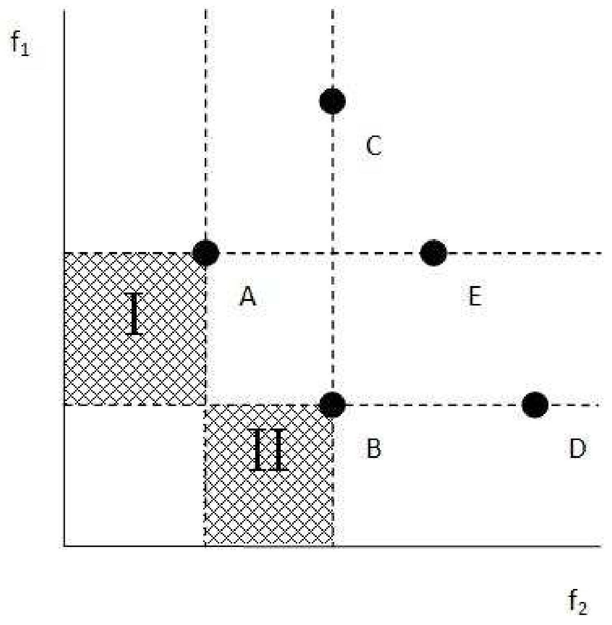


Figure 2.3: Illustration Pareto optimal points

In words, this definition says that $\boldsymbol{x}^*$ is Pareto optimal if there exists no feasible vector $\boldsymbol{x}$ which would decrease some objective without causing a simultaneous increase in at least one other objective. This concept gives not a single solution, but rather a set of solutions called the Pareto optimal set or Pareto front.

This can be illustrated with the following example. In figure 2.3 several points are plotted against two objectives functions. In this example, points A and B are Pareto optimal. For both points, there is no other point which has a smaller or equal value on both objective functions and smaller for at least one, i.e. no points in subspaces I and II. For all other points, there exists a point that has a smaller or equal value for each objective function.

The vectors corresponding to the solutions included in the Pareto optimal set are so-called non-dominated points.

Chapter 3

Metamodels

In product design and optimisation studies, metamodels are used to approximate the design analysis and provide a model of a model [17]. In this study the metamodels are not considered as a surrogate model, such as a low-fidelity physics code, but more as a mathematical approximation of the computer simulation or analysis [31]. The design analysis is considered as a black-box, so for every design point only the vector of input variables $\boldsymbol{x}$ (properties/design) and the vector of output variables $\boldsymbol{y}$ (characteristic/performance) are needed.

For simplicity in the following, the output variable is taken as an one-dimensional, scalar output y. Hence, for models with several output variables, a separate metamodel is fit to each output variable.

Let the functional relation $f : \mathbb{R}^n \rightarrow \mathbb{R}$ between $\boldsymbol{x}$ and y be expressed as:

$$y = f(\boldsymbol{x}). \tag{3.1}$$

Also, let there be a dataset $(\boldsymbol{x}^{(i)}, y^{(i)})$ consisting of inputs for experimental runs of the design analysis and its responses. Each row $i = 1, \ldots, N$ specifies a design $\boldsymbol{x}$ based on input variables $x_1, \ldots, x_v$. Let $\boldsymbol{y}$ be a vector of output responses, corresponding with the specified designs, with each row $i = 1, \ldots, N$ containing the performance measures y of the output response. The metamodel can be written as

$$\hat{y} = \hat{f}(\boldsymbol{x}), \tag{3.2}$$

where $\hat{y}$ is the response variable predicted by the metamodel and $\hat{f}(\cdot)$ is the mathematical function used by the metamodel.

In general, the metamodel does not approximate the design analysis precisely, so an error will be involved. In other words, the predicted response value $\hat{y}$ will differ from the observed response variable y. Let an error be defined by $\varepsilon = y - \hat{y}$. This yields the following functional representation of the metamodel

$$y = \hat{f}(\boldsymbol{x}) + \varepsilon. \tag{3.3}$$

The design analyses considered in this study are deterministic, i.e. there is no random error involved in the design analyses. However, some of the metamodels, for example stepwise regression [57], employ statistical methods to build the approximation. Therefore it should be noted that for deterministic analyses "usual measures of uncertainty derived from least-squares residuals have no obvious statistical meaning" [40]. Of course, least-squares regression can be viewed as curve fitting and thus used, but statistical measures to verify model adequacy have no meaning, since they assume a random error term. The error ε in equation 3.3 represents only the error of approximation.

Roughly speaking, metamodels can be subdivided in two categories, namely interpolating and approximating metamodels. An interpolating metamodel 'honours' the data and the prediction of the known datapoints will be exact, i.e. there will be no error. An approximating metamodel, however will smooth the datapoints and the prediction of the known datapoints are not necessary exact.

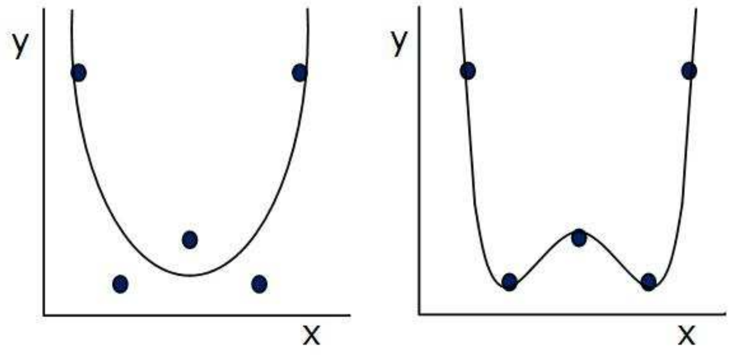


Figure 3.1: One-dimensional illustration of approximating (left) and interpolating (right) metamodels

An one-dimensional example of approximating and interpolating metamodels is given in figure 3.1. The lines represent the metamodels, and the points the known data points. The approximating metamodel predicts some of the data points exactly, while the interpolating metamodel predicts every known data point exactly.

In metamodelling, many different types of mathematical functions are used for describing the function $\hat{f}$ in equation 3.2. The following sections will describe the different types of metamodels that are considered in this study.

3.1 Polynomial models

Polynomial models determine the function of the metamodel $\hat{f}(\boldsymbol{x})$ through a systematic decomposition of the variability in the observed response values [11]. The polynomial coefficients are then estimated by minimizing the aboslute value of the error ε. Note that the dataset is assumed to be deterministic, so the error of approximation is not due to random effects.

Polynomial models can be of any order. The most widely used metamodels are first and second order polynomials [43]. The general form of a polynomial model of order n is

$$\hat{y} = \beta_0 + \sum_{1\le i_1 \le v} \beta_{i_1} x_{i_1} + \sum_{1\le i_1 \le i_2 \le v} \beta_{i_1,i_2} x_{i_1} x_{i_2} \\ + \ldots + \sum_{1\le i_1\le i_2 \le \ldots \le i_n \le v} \beta_{i_1,\ldots,i_n} x_{i_1} \cdots x_{i_n}, \quad (3.4)$$

where $\beta.$ are the unknown polynomial coefficients to be estimated, and x_{i_j} are the explanatory variables, where $i_j = 1, \ldots, v$ with v the number of explanatory variables, and $\hat{y}$ is the predicted response value.

The number of these coefficients $\beta.$ and thus the number of terms in the polynomial model are determined by the order n of the polynomial and the number v of explanatory variables. The number of coefficients is given by

$$\sum_{i=1}^{\min\{n,v\}} \binom{v}{i} \binom{n}{i}. \quad (3.5)$$

Least-squares regression [33] is typically used to estimate the vector of regression coefficients $\boldsymbol{\beta}$ in a polynomial model [42]. The polynomial model equation 3.4 can be written in matrix notation as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad (3.6)$$

where the so-called design matrix $\boldsymbol{X}$ is given by

$$\begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_v^{(1)} & x_1^{(1)}x_2^{(1)} & \cdots & x_{v-1}^{(1)}x_v^{(1)} & \left(x_1^{(1)}\right)^2 & \cdots & \left(x_v^{(1)}\right)^2 & \cdots \\ 1 & x_1^{(2)} & \cdots & x_v^{(2)} & x_1^{(2)}x_2^{(2)} & \cdots & x_{v-1}^{(2)}x_v^{(2)} & \left(x_1^{(2)}\right)^2 & \cdots & \left(x_v^{(2)}\right)^2 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \cdots \\ 1 & x_1^{(N)} & \cdots & x_v^{(N)} & x_1^{(N)}x_2^{(N)} & \cdots & x_{v-1}^{(N)}x_v^{(N)} & \left(x_1^{(N)}\right)^2 & \cdots & \left(x_v^{(N)}\right)^2 & \cdots \end{bmatrix}, \tag{3.7}$$

where $\boldsymbol{x}_i$ for $i = 1, \ldots, v$ represents the vector of the corresponding explanatory variable found through the experimental runs.

To estimate the unknown coefficients $\boldsymbol{\beta}$ of the polynomial model the least squares method is used. The least squares method estimates the vector of regression coefficients $\boldsymbol{\beta}$ by minimizing the sum of the squares of the residual vector $\boldsymbol{\varepsilon}$. This is done by solving the least squares normal equations, which coincides with:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}. \tag{3.8}$$

3.1.1 Collinearity

However, for $\boldsymbol{X}'\boldsymbol{X}$ to be invertible, the matrix $\boldsymbol{X}$ must be of full rank. A matrix is of full rank if all its columns are linearly independent. A linear relationship among the columns of $\boldsymbol{X}$ is called collinearity. If $\boldsymbol{X}$ does not have full rank, which is called rank deficient, the solution to the least squares problem is not unique. This means, that there are infinitely many vectors of regression coefficients $\hat{\boldsymbol{\beta}}$ that solve the least squares normal equations, given in equation 3.8. Note that all regression coefficients yield different predictions. However, the presence of collinearity does not affect the accuracy of the prediction, i.e. the sum of squares of each of the infinitely many residual vectors is equal.

In high-order polynomial models collinearity can occur in the design matrix. This is caused by the existence of a correlation between different powers of the same or different variable(s). Schacham and Brauner [41] conclude that a $z[-1, 1]$ transformation minimizes the effects of collinearity. This $z[-1, 1]$ transforms each explanatory variable x_i onto the interval $[-1, 1]$ by the following transformation:

$$z_i = \frac{2x_i - x_{\max} - x_{\min}}{x_{\max} - x_{\min}}, \tag{3.9}$$

where $x_{\max}$ and $x_{\min}$ is determined for each explanatory variable.

Alternatively, Bradley and Srivatava [4] recommend adjusting the origin of the explanatory variables so that the mean of the values used is zero ($x_i - \bar{x}$), also called centering. This removes correlation between even and odd powers. However, these even and odd powers can still be highly correlated with other like powers.

The explanatory variable which causes the collinearity can also be neglected. This creates a smaller model with less coefficients and terms. However, this is not advisable, since dropping an explanatory variable also eliminates a lot of (probably) valuable information.

Eventually, the recommended way to reduce or even remove collinearity is by adding more observations to the dataset. Additional data points that provide independent variation relative to the original data can be useful. However, there is no guarantee that the additional data points will provide independent information.

3.2 Kriging

A kriging model is a generalized linear regression model that accounts for the correlation in the residuals between the regression model and the observation values [12]. Given the mathematical form of kriging, the process of using kriging first requires the estimation of the 'best' parameters, and an assessment of the resulting kriging model's accuracy before it can be used as an approximation to a deterministic computer model.

The mathematical form of a kriging model has two parts:

$$y = \boldsymbol{\beta f}(\boldsymbol{x}) + Z(\boldsymbol{x}) \tag{3.10}$$

The first part, $\boldsymbol{\beta f}(\boldsymbol{x})$, is a linear regression of the data modeling the drift of the process mean, also called the 'trend' over the domain. The second part $Z(\boldsymbol{x})$ is a realization of a stochastic process with mean zero, variance σ^2 and non-zero covariance. Most previous engineering applications utilize a constant trend model over the domain [42] and rely on the second part of the model to 'pull' the regression model through the observed data by quantifying the correlation of nearby points.

The general form of kriging is defined with a vector of regression functions

$$\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})]^T. \tag{3.11}$$

A first-order linear regression model would be given by the following vector of regression functions

$$\boldsymbol{f}(\boldsymbol{x}) = [1, \boldsymbol{x_1}, \ldots \boldsymbol{x_v}]^T. \tag{3.12}$$

Ordinary kriging, utilizing a constant regression function given by

$$\boldsymbol{f}(\boldsymbol{x}) = [1, \ldots, 1]^T, \tag{3.13}$$

which yields the kriging model

$$y = \beta + Z(\boldsymbol{x}), \tag{3.14}$$

is the most commonly used form of kriging employed to approximate computer models [40], [43], [3], [53], [7], [19].

The response values at known data points are predicted exactly. If an unobserved point, x, moves away from the observations, the second component of the kriging model approaches zero, yielding the generalized least-squares estimate.

The second part $Z(\boldsymbol{x})$ is a model of a Gaussian and stationary random process with zero mean and covariance:

$$V(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \sigma^2 R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}), \tag{3.15}$$

where $\boldsymbol{x}^{(i)}$ is the i-th observation of the input data.

The variance of the random process is denoted by σ^2, which can be determined from the fact that $R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}) = 1$. The process variance also acts as a scalar of the spatial correlation function (SCF), given by $R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$. The SCF controls the smoothness of the resulting kriging model, the influence of nearby point, and the differentiability of the surface by quantifying the correlation between observations.

Koehler et al. [19] provide an overview of four common SCF's used for approximating a deterministic computer model and describe the impact of the selection of different parameter values for these functions. However, the correlation function $R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ can also be specified by the user, and a variety of correlation functions exists [19], [32], [40].

If $\boldsymbol{x}^{(i)}$ is multi-dimensional, the correlation of each dimension is treated apart, which is mathematically noted by:

$$R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \prod_{k=1}^{v} R_k(\boldsymbol{x}_k^{(i)}, \boldsymbol{x}_k^{(j)}). \tag{3.16}$$

More specific, the correlation functions in table 3.1 are utilized, which are taken from [25].

The cubic spline correlation model is defined by

$$\varsigma(\xi) = \begin{cases} 1 - 15\xi^2 + 30\xi^3 & \text{for } 0 \leq \xi \leq 0.2, \\ 1.25(1-\xi)^3 & \text{for } 0.2 < \xi < 1, \\ 0 & \text{for } \xi \geq 1. \end{cases} \tag{3.17}$$

SCF Name	SCF expression $R_k(\boldsymbol{x}_k^{(i)}, \boldsymbol{x}_k^{(j)})$	
Exponential	$\exp\left(-\theta_k \left\lvert \boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)} \right\rvert\right)$	
Exponential/Gaussian	$\exp\left(-\theta_k \left\lvert \boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)} \right\rvert^d\right)$,	$0 < d \leq 2$
Gaussian	$\exp\left(-\theta_k \left(\boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)}\right)^2\right)$	
Linear	$\max\left\{0, 1 - \theta_k \left\lvert \boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)} \right\rvert\right\}$	
Spherical	$1 - 1.5\xi_k + 0.5\xi_k^3$,	$\xi_k = \min\left\{1, \theta_k \left\lvert \boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)} \right\rvert\right\}$
Cubic	$1 - 3\xi_k^2 + 2\xi^3$,	$\xi_k = \min\left\{1, \theta_k \left\lvert \boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)} \right\rvert\right\}$
Cubic Spline	$\varsigma(\xi_k)$,	$\xi_k = \theta_k \left\lvert \boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)} \right\rvert$

Table 3.1: Available spatial correlation functions

The spatial correlation function range parameter θ has little meaning in a physical sense. A different θ can be used for each design variable, which yields the vector $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_v]$. However, in some cases, using a single correlation parameter for all design variables gives sufficiently good results [40], [34], [2]. This is also described as isotropic, this means that one correlation is identified in different directions. Consequently, a vector $\boldsymbol{\theta}$ is called anisotropic.

The correlation matrix $\boldsymbol{R}$ is composed of spatial correlation functions evaluated at each possible combination of the known points:

$$\boldsymbol{R} = \begin{bmatrix} R(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}) & R(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) & \cdots & R(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(N)}) \\ R(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}) & R(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(2)}) & \cdots & R(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(N)}) \\ \vdots & \vdots & \ddots & \vdots \\ R(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(1)}) & R(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(2)}) & \cdots & R(\boldsymbol{x}^{(N)}, \boldsymbol{x}^{(N)}) \end{bmatrix} \tag{3.18}$$

This matrix $\boldsymbol{R}$ is a positive semi-definite matrix because the SCF defining each element is positive semi-definite [28]. It is also symmetric because $R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = R(\boldsymbol{x}^{(j)}, \boldsymbol{x}^{(i)})$, and the diagonal consists of all ones because $R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}) = 1$.

The correlation between an unknown point $\boldsymbol{x}^*$ and the N known sample points is given by this vector

$$\boldsymbol{r}(\boldsymbol{x}^*) = \left[R(\boldsymbol{x}^*, \boldsymbol{x}^{(1)}), R(\boldsymbol{x}^*, \boldsymbol{x}^{(2)}), \ldots, R(\boldsymbol{x}^*, \boldsymbol{x}^{(N)})\right]^T \tag{3.19}$$

The best linear unbiased predictor (BLUP) is obtained by minimizing the mean square error of the predictions. This leads to the following BLUP of an unknown point $\boldsymbol{x}^*$

$$\hat{y}^* = \boldsymbol{f}^T(\boldsymbol{x}^*)\hat{\boldsymbol{\beta}} + \boldsymbol{r}^T(\boldsymbol{x}^*)\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}^T), \tag{3.20}$$

where the column-vector $\boldsymbol{F}$ is constructed by evaluating $\boldsymbol{f}(\boldsymbol{x})$ at each of the N known observations.

In the area of design and analysis of computer experiments [40], the statistics-based method of maximum likelihood estimation (MLE) is primarily used as an objective estimator of the best kriging model parameters (β, θ and σ) that are most consistent with the observed data [7], [26], [16]. MLE assumes the residuals have a known probability distribution shape, which in most cases is the Gaussian probability distribution. Stein [46] argued that estimation of the parameters through general cross-validation (GCV), an alternative option for estimation, will employ twice the variance of the maximum likelihood estimation. But Wahba [51] still advocated GCV, because supposedly more robustness against departures from the stochastic model. Martin and Simpson [28] concluded that MLE was the best method to estimate kriging model parameters even if the modelled observations do not have a Gaussian distribution. Cross validation has the potential of performing slightly better, especially for a constant trend function, but more importantly has the potential of performing much worse.

The MLE estimation of β matches its least-squares estimate and is given by

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{y}, \tag{3.21}$$

and the estimation of σ^2 is given by

$$\hat{\sigma}^2 = (1/n)(\boldsymbol{y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}^T)^T\boldsymbol{R}^{-1}\boldsymbol{y} - \boldsymbol{F}\hat{\boldsymbol{\beta}}^T). \tag{3.22}$$

3.3 Radial Basis Functions

Radial basis functions (RBF) have been developed for scattered multivariate data interpolation [14], [8]. The method uses linear combinations of a radially symmetric function based on Euclidean distance or other such metric to approximate response functions [8], [37]. Mathematically, the model can be expressed as

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{N} w_i\phi\left(||\boldsymbol{x} - \boldsymbol{x}^{(i)}||\right), \tag{3.23}$$

where $\phi(\cdot)$ denotes the radial basis functions, $||\cdot||$ denotes the Euclidean norm, the given samples $\boldsymbol{x}^{(i)}$ for $i = 1, \ldots, N$ are the centres of the radial-basis functions, and w_i for $i = 1, \ldots, N$ are unknown coefficients.

The coefficients w_i are found by replacing the left hand side of the equation with the real responses $\boldsymbol{y}$ and solving the resulting linear system

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{w}, \tag{3.24}$$

where $\boldsymbol{w}$ is the vector of the to be estimated weights and the coefficients a_{ij} of the matrix $\boldsymbol{A}$ can be obtained from

$$a_{ij} = \phi\left(||\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}||\right). \tag{3.25}$$

The most commonly used radial basis functions are linear, thin-plate spline, Gaussian, and multi-quadratic functions [9]. Recently researchers have developed new basis functions; representative examples are the compactly supported basis functions developed by Wu [56] and Wendland [54]. In this study only the Gaussian radial basis function will be considered. The mathematical representation of this Gaussian radial basis function is

$$\phi(||\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}||) = e^{-c||\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}||^2}, \tag{3.26}$$

where c denotes the width parameter of the Gaussian function. The width parameter of the Gaussian function determines the width of the bell-shaped Gaussian function and thereby the influence of nearby points. A large width means that point further away from an unknown points will have an effect on the prediction at that point. A small value means that only nearby points will have an effect.

As can be seen in the figure 3.2, over- and underfitting can occur. If the value of the width parameter is taken too small, overfitting will occur, i.e. every point will only have an influence in the very nearby neighbourhood. On the other hand, if the width parameter is taken too large, generalization will occur, and all points will be regarded as one point.

RBF can model high-order non-linear responses well, but are inappropriate for linear and quadratic responses [22]. Radial basis function approximations have been shown to produce good fits to arbitrary contours of both deterministic and stochastic response function [37]. Tu and Barton [47] found that RBF approximations provide effective metamodels for electronic circuit simulation models. Meckesheimer et al. [30] used the method for constructing metamodels for a desk lamp design example, which has both continuous and discrete response functions.

3.4 Support Vector Regression

Support vector regression (SVR) is a particular implementation of support vector machines (SVM), what is “a principled and very powerful method that

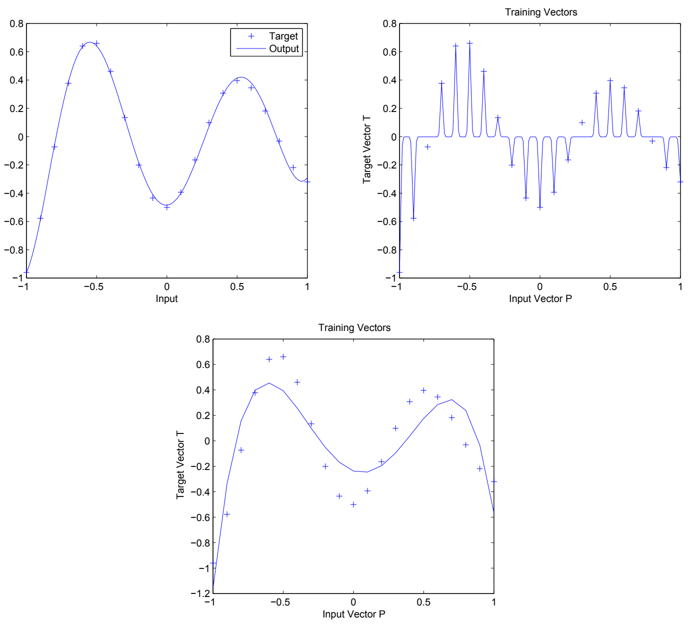


Figure 3.2: Over- and underfitting by varying the width parameter of the Gaussian radial basis function

in the few years since its introduction has already outperformed most other systems in a wide variety of applications" [6]. The SVM algorithm is a non-linear generalization of the generalized portrait algorithm developed in Russia in the 1960's [49]. In its present form, SVM was developed by Vapnik in the early 90's [50]. Smola et al. [44] acknowledge the success of SVM's since this time and also add that "in regression and time series prediction applications, excellent performances were soon obtained." The resulting support vector regression is showing promising empirical performance [48], [13].

A SVR can be represented by the typical mathematical function

$$\hat{f}(\boldsymbol{x}) = \boldsymbol{w} \cdot \Phi(\boldsymbol{x}) + \boldsymbol{b}, \tag{3.27}$$

where Φ denotes a certain transformation of $\boldsymbol{x}$, and the vectors $\boldsymbol{w}$ and $\boldsymbol{b}$ are the to be estimated parameters.

The parameters $\boldsymbol{w}$ and $\boldsymbol{b}$ are estimated with the so-called ε-insensitive loss function [13]. A function $\hat{f}(\boldsymbol{x})$ is searched so that the dataset $(\boldsymbol{x}^{(i)}, y^{(i)})$ can be approximated with ε precision. This means that the prediction of the SVR metamodel, $\hat{y}_i$, of each data point may differ at most ε with the real value y_i.

However, slack variables, ξ_i and ξ_i^*, are also incorporated into the optimisation problem, because a function with ε precision may not exist.

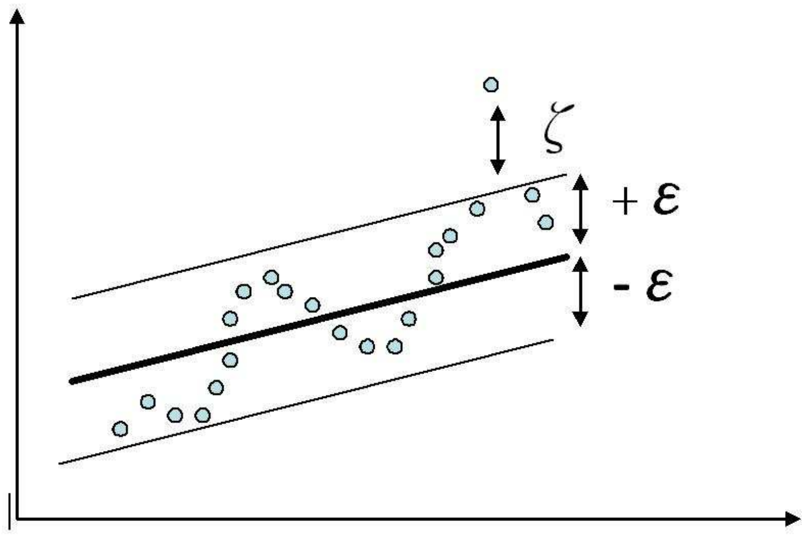


Figure 3.3: Slack variables and the ε-insensitive loss function in SVR [55]

The idea is illustrated in figure 3.3 and the optimisation problem is given by:

$$\min \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \tag{3.28}$$

subject to

$$\begin{cases} y_i - \boldsymbol{w}\cdot\Phi(\boldsymbol{x}) - \boldsymbol{b} \leq \varepsilon + \xi_i \\ \boldsymbol{w}\cdot\Phi(\boldsymbol{x}) + \boldsymbol{b} - y_i \leq \varepsilon + \xi_i^* \\ \varepsilon, \xi_i, \xi_i^* \geq 0 \end{cases} \tag{3.29}$$

where the constant $C > 0$ determines the costs for each deviation that is larger than ε.

The optimisation function and linear constraints can be written as the Lagrangian function, which yields the optimisation problem in dual form

$$\max -\frac{1}{2}\sum_{i,j=1}^{l} Q_{ij}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) - \varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + y_i\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) \tag{3.30}$$

subject to

$$\begin{cases} \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \\ (\alpha_i - \alpha_i^*) \in [0, C], \end{cases} \tag{3.31}$$

where $Q_{ij} = \Phi(\boldsymbol{x}^{(i)})\Phi(\boldsymbol{x}^{(j)}) = k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ representing the kernel function.

The input vectors only appear inside the kernel functions. In this way, the dimensionality of the input vectors is hidden from the remaining computations, providing means for addressing the curse of dimensionality [13].

The following kernel functions are considered in this study

Linear	$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x} \cdot \boldsymbol{x}' \rangle$
Polynomial	$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x} \cdot \boldsymbol{x}' \rangle^d$
Gaussian	$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\lVert \boldsymbol{x} - \boldsymbol{x} \rVert^2}{2\sigma^2}\right)$
Exponential	$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\lVert \boldsymbol{x} - \boldsymbol{x}' \rVert}{2\sigma^2}\right)$
Sigmoid	$k(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\rho \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \theta)$
Fourier	$k(\boldsymbol{x}, \boldsymbol{x}') = \frac{\sin(N + \frac{1}{2})(\boldsymbol{x} - \boldsymbol{x}')}{\sin(\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}'))}$
Splines	$k(\boldsymbol{x}, \boldsymbol{x}') = 1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \frac{1}{2} \langle \boldsymbol{x}, \boldsymbol{x}' \rangle \min(\boldsymbol{x}, \boldsymbol{x}') - \frac{1}{6} \min(\boldsymbol{x}, \boldsymbol{x}')^3$
B-Splines	$k(\boldsymbol{x}, \boldsymbol{x}') = B_{2N+1}(\boldsymbol{x} - \boldsymbol{x}')$

Table 3.2: Kernel functions of the SVR metamodel

Finally, the primal form of the optimisation yields the following function $f(\boldsymbol{x})$ for the SVR

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) k(\boldsymbol{x}^{(i)}, \boldsymbol{x}) + b. \tag{3.32}$$

3.5 Multivariate Adaptive Regression Splines

Multivariate adaptive regression splines (MARS) is a non-parametric regression that does not take a predetermined form but is constructed according to information derived from the data. MARS constructs the functional relation from a set of coefficients and basis functions that are determined from regression data [10]. The basis functions for approximating the response function are adaptively selected through a forward and a backward iterative approach [10].

The MARS metamodel can be described by its basis functions B_i and coefficients a_i for $i = 1, \ldots, M$ noted as

$$\hat{\boldsymbol{y}} = \sum_{m=1}^{M} a_m B_m(\boldsymbol{x}). \tag{3.33}$$

The coefficients are here estimated through least-squares regression of the basis functions $B_m(\boldsymbol{x})$ to the observed values $\boldsymbol{y}$.

First, a forward iterative approach is utilized which selects the model terms in the MARS metamodel. The maximum number of model terms can be user-specified and is noted by $M_{\max}$. After the forward iterative approach is done, a backward iterative approach is started. This backward iterative approach considers subsets of the model terms selected by the forward iterative approach and finds the best subset. The best subset is determined by its lack-of-fit, which is represented by the least-squared criterion.

The MARS model terms are based on truncated linear functions. These truncated linear functions are typically noted by

$$b^{+}(x_v - k) = \left[+(x_v - k)\right]_{+}, \quad b^{-}(x_v - k) = \left[-(x_v - k)\right]_{+}, \tag{3.34}$$

where $[q]_{+} = \max\{0, q\}$. These functions are thus truncated to zero at a knot k. The knots determine where the approximation bends to model curvature. Basically, both functions are added to the MARS metamodel as model terms.

These truncated functions mentioned before are univariate, but the MARS metamodel also incorporates interaction basis functions. All basis functions (univariate and interaction) can be mathematically represented by

$$B_m(\boldsymbol{x}) = \prod_{l=1}^{L_m} \left[s_{l,m} \cdot (x_{v(l,m)} - k_{l,m})\right]_{+}. \tag{3.35}$$

K_m denotes here the number of interaction terms of basis function m, and $s_{l,m}$ the direction in which the knot is pointed (+ or -). Note that also self-interactions can exist.

The selection of basis functions and knots stops when a certain number, $M_{\max}$, is reached. These number is user-specified and provide a trade-off between computational time and interpolation character.

The backward stepwise part starts with all $M_{\max}$ basis functions derived from the forward stepwise algorithm. It omits one basis function at a time and finds the best set of basis functions for the MARS approximation.

3.6 Neural Networks

A neural network is composed of neurons which are multiple linear regression models with a non-linear transformation on y. If the input to each neuron are denoted by the explanatory variables $\boldsymbol{x}$ and the regression coefficients are denoted by the weights $\boldsymbol{w}$, then the output y is given by the hyperbolic tangent sigmoid transfer function

$$y = \frac{2}{1 + e^{-2\eta}} - 1, \tag{3.36}$$

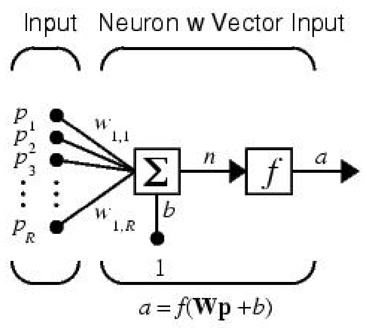


Figure 3.4: Outline of a single neuron

where $\eta = \boldsymbol{w}^T \boldsymbol{x} + \beta$, where β is the bias value of a neuron.

A neural network is then created by assembling the neurons into an architecture; in this study we consider the multilayer feed-forward architecture. Feed-forward layered networks have the flexibility to approximate smooth functions arbitrarily well, provided sufficient nodes and layers. This follows from the work of Kolmogorov [21], whose results imply that any continuous function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ can be exactly reproduced over a compact subset by a three-layer feed-forward network, as seen in the following figure.

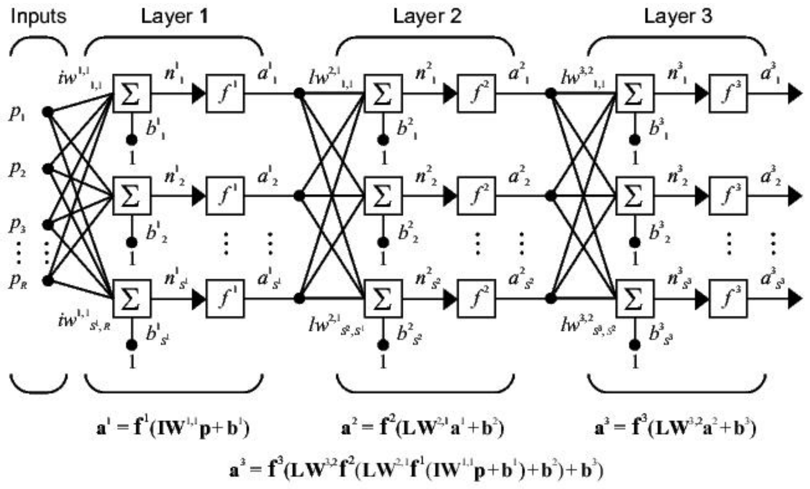

Inputs
Layer 1
Layer 2
Layer 3
$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1)$
$\mathbf{a}^2 = \mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{a}^1 + \mathbf{b}^2)$
$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{a}^2 + \mathbf{b}^3)$
$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3)$

Figure 3.5: Architecture of a neural network

There are two main issues in building a neural network:

1. specifying the architecture,

2. training the neural network to perform well with reference to a training set.

If the architecture is made large enough, a neural network can be a nearly universal approximator [38]. However, more hidden layers will result in more connections so that the computational costs increases strongly. Training a neural network is the determination of the proper values for all weights in the architecture and is done by back-propagation [38].

Neural networks are best suited for approximating deterministic functions in regression-type applications [43]. “In most applications of neural networks that generate regression-like output, there is no explicit mention of randomness. Instead, the aim is function approximation.” [5]

Chapter 4

Metamodel assessment

Metamodel assessment is considered in this study as the judgement of the quality or fidelity of a metamodel, where the quality or fidelity must be evaluated quantitatively [31]. The assessment of metamodels also provides valuable information for metamodel improvement.

An approximation is a value that is close to the intended value but is in general not equal. When building metamodels for the purpose of design optimisation, there are three important aspects to take into account [31]:

1. obviously, building a good approximation,
2. generate measures of performance to assess the goodness of the approximation, and
3. provide an indicator of confidence for the estimated measures of performance.

The first aspect makes studying a (complex or computationally expensive) underlying function or model more rapidly possible. This involves the choice of an appropriate metamodel type and form for constructing a simplified model of the underlying function or model. However, this raises the issue of defining the meaning of 'good' when using metamodels for engineering design.

The second aspect provides measures of performance to assess the loss of information or accuracy in the metamodel that is traded off against the increase in speed of analysis. The goodness of a metamodel may not be dictated by a single performance measure but could depend on several different measures, depending on its intended use [15]. Initially, designers may be looking for an indication of useful domains of the design variables and the identification of key variables. During optimisation, designers may be interested in obtaining measures of performance for assessing the impact of design constraints on the optimal objective, refining optimisation formulations, and determining globally optimal designs. Finally, designers may desire measures of performance for evaluating trade-offs in the presence of competing objectives to determine the adequacy of their solutions. Therefore, it is important that these measures of performance be relevant and informative to the user for judging whether the metamodel is acceptable for its intended purpose.

Finally, the third aspect provides an indicator of confidence for the measures of performance when they are estimated. The issue of providing an indicator of confidence for the estimated performance measures is an area where further research is needed.

Validation is necessary whenever a metamodel is meant to represent a underlying model. Kleijnen and Sargent [18] define validation as the "... verification that a model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model."' Validation relates to both the metamodel and the underlying model and requires knowledge about the problem and the specified accuracy required of the metamodel.

Basically, the accuracy of a metamodel can be determined by the residuals of the metamodel. A metamodel with small residuals can be considered to represent the dataset more accurately then a metamodel with high residuals. This way the quality of metamodels can be assessed and several metamodels can be compared with this criterion.

However, some metamodels have a interpolating character, which means that no residuals are present. These metamodels give exact prediction of known data points. It is impossible to base the quality of the metamodel on the error of predictions with these metamodels.

A generic form of assessing metamodel quality and validation is called cross-validation. With cross-validation interpolating metamodels can be compared to non-interpolating metamodels. Cross-validation will be discussed in the next section.

4.1 Cross-validation

Cross-validation is a method for estimating the error of prediction of a metamodel (e.g. Meckesheimer et al. [29], [31]). Cross-validation is also helpful in the process of metamodel selection, because the cross-validation errors of each metamodel can be compared and the metamodel with the smallest error can be selected. This validation method provides also more insight in the relevance of input variables to the accuracy of the metamodel. Here, an estimation of a statistical measure of performance, an estimated cross-validation error measure, to provide an assessment of the fidelity of a metamodel, is considered.

Basically, cross-validation uses one dataset for both fitting and validating a metamodel. The most simple form of cross-validation splits a dataset into two parts. The first part is used for fitting the metamodel and the second part is used for computing the cross-validation prediction errors. The two parts are then switched and the second part of the dataset is used for fitting, whereas the data points in the first part will now be predicted. This switching step is the characteristic step for cross-validation.

However, the data need not necessarily be split into two parts, and other schemes for randomizing and partitioning a data set may be used as discussed by Laslett [23].

Splitting a dataset into two subsets is also called 2-fold cross-validation. Generically, this method is called p-fold cross-validation. The dataset is split into p mutually exclusive and exhaustive subsets. Then, the metamodel is fit p times, each time omitting one of the subsets from fitting and using the omitted subset to compute the cross-validation error measure.

With N-fold cross-validation, where N is the number of data points in the dataset, the dataset is split into N subsets, each subset consisting of one data point. This is a special variation of p-fold cross-validation, called leave-one-out cross-validation. Each time one data point is left out of the fitting set, and this point will be predicted by the metamodel fitted on all other points. Mitchell and Morris [32] describe how the cross-validation error measure may be computed inexpensively for leave-one-out cross-validation.

Typically, all possible $\binom{N}{k}$ subsets of size k can be left out, and the metamodel is fit to each remaining set. Each time, the cross-validation error measure is computed at the omitted points. This is called leave-k-out cross-validation. This approach is a computationally more expensive version of p-fold cross validation.

Based on the observations from the experimental study conducted to assess the leave-k-out strategy [31], a value of $k = 1$ is suggested for providing a prediction error estimate for radial basis functions and low-order polynomials metamodels but not for kriging metamodels. Choosing k as a function of the fitting design size (that is, $k = 0.1$ or $k = \sqrt{N}$) was instead recommended for estimating the prediction error for kriging metamodels.

Lin [24] found through intensive testing that the leave-one-out cross-validation is an insufficient measurement for metamodel accuracy. The leave-one-out cross-validation is actually a measurement for degrees of insensitivity of a metamodel to lost information at its data points, while an insensitive metamodel is not necessarily accurate. A "validated" model by leave-one-out could be far from the actual as the data points may not be able to capture the actual. Designers are in danger of accepting an inaccurate metamodel that is insensitive to lost information at data points, and inaccurate and insensitive metamodels might be the result of poor distribution of the data points. On the other hand, with leave-one-out cross-validation there is some danger of rejecting an accurate metamodel that is also sensitive to lost information at data points.

It has to be noted that although cross-validation is a limited measurement for metamodel accuracy, it still is favourable. When dealing with interpolating metamodels, quality assessments must be made on unknown data points. At the same time, all available information is desired to incorporate into the metamodel. Cross-validation has these two characteristics, whereas other methods don't. Because assessing metamodel accuracy is essential in the metamodelling process, leave-one-out cross-validation is often chosen as validation strategy, because of its ease of use.

4.2 Accuracy metrics

To express the value or importance of the errors (or residuals) in cross-validation assessments, there are different metrics available in the literature [15], [52]. An overview is given below.

Root Mean Square Error (RMSE) / Mean Square Error (MSE)

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \tag{4.1}$$

$$\text{MSE} = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N} \tag{4.2}$$

The lower the value of RMSE, the more accurate the metamodel. RMSE is used to gauge the overall accuracy of the model.

R-Square

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \tag{4.3}$$

where $\hat{y}_i$ is the corresponding predicted value for the observed value y_i, and $\bar{y}$ is the mean of the observed values. It also can be seen as the mean square error (MSE) divided by the variance. While the MSE represent the departure of the metamodel from the real simulation model, the variance capture how irregular the problem is. The larger the value of R-Square, i.e. closer to 1, the more accurate the metamodel.

Relative Average Absolute Error (RAAE) / Average Absolute Error (AAE)

$$\text{AAE} = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N}, \tag{4.4}$$

$$\text{RAAE} = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N\sigma}, \tag{4.5}$$

where σ stands for standard deviation. The smaller the value of RAAE, the more accurate the metamodel.

Relative Maximum Absolute Error (RMAE) / Maximum Absolute Error (MAE)

$$\text{RMAE} = \frac{\max\{|y_1 - \hat{y}_1|, \ldots, |y_N - \hat{y}_N|\}}{\sigma} \tag{4.6}$$

$$\text{MAE} = \max\{|y_1 - \hat{y}_1|, \ldots, |y_N - \hat{y}_N|\} \tag{4.7}$$

While the RAAE is usually highly correlated with MSE and thus R-Square, RMAE is not necessarily. Large RMAE indicates large error in one region of the design space even though the overall accuracy indicated by R-Square and RAAE can be very good. Therefore, a small RMAE is preferred.

Mean Absolute Percentage Error (MAPE)

$$\text{MAPE} = \frac{\sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{y_i}}{N} \times 100\% \tag{4.8}$$

The MAPE correlates with the AAE, but each error is divided by its true value.

The most commonly used error metric is RMSE. This error metric is represented in the same dimension as the dataset, i.e. the value of the RMSE does not have to be scaled. The value of the RMSE represents the mean error, and due to the squared value large errors are given more weight than small errors. The unscaled variant of the RMSE is the MSE, which is simply its square.

The RMSE can be biased, because the errors are not relatively measured. Errors over large and small values are not weighted, while it can be desirable, e.g. with a dataset with high and low observation values, that a small error on a small value is equally accounted as a larger error on a large error. MAPE takes this aspect into account. It measures the error relatively to the real values. This yields a the proportion of the real value that equals the error. The MAPE therefore represents the mean error in terms of a percentage of the real values.

R-Square gives also, like MAPE, a percentage representing the accuracy of the model. This error metric is highly correlated with and a scaled version of the MSE. The R-Square captures the irregularity of the dataset by dividing the MSE by the variance of the dataset. The metric is very understandable, like MAPE, because percentages are very tangible.

The last error metric that is considered is the (R)MAE. Unlike the other error metrics, that consider the whole dataset, this error metric can identify locally large errors. This error metric displays the largest error of approximation. Other error metrics take an average value, whereas this error metric shows a solely value. Locally inaccurate metamodels can hereby be identified.

Chapter 5

Implementation

The metamodels discussed before are applied to datasets. At NLR a program was developed in MATLAB by the NLR that can calculate these metamodels. This program is called MultiFit and is comparable to the toolboxes provided together with MATLAB. MultiFit is a tool for fitting metamodels on given datasets, calculating predictions of unknown data points and providing qualitatively accuracy measurements of metamodels. MultiFit consists among others, of several MATLAB functions, which are run from the command-line of MATLAB. The main functions of the MultiFit program will be discussed in the following sections.

5.1 Fitting metamodels

The MATLAB function that is responsible for fitting and calculating the actual metamodels is called `MF_Fit`. Given a dataset and a desired metamodel, `MF_Fit` fits the metamodel at the dataset. Furthermore, the function calculates the coefficients of the metamodel and provides a model which can be evaluated for predicting other data points.

5.1.1 Usage of the function

The syntax of the `MF_Fit` function can be described as:

```
models = MF_Fit(Xfit,Yfit,method,param)
```

`MF_Fit` takes as arguments the dataset that will be fitted, provide by the explanatory variables `Xfit` and the response variables `Yfit`. The columns of `Xfit` represent the different explanatory variables and the rows of `Xfit` represent the experiments. Consequently, the columns of `Yfit` represent the different response variables. Note that multiple response variables can be passed to `MF_Fit`, which makes it a multi-objective fitting tool. However, a single metamodel is calculated for each response variable. The rows of the `Xfit` and `Yfit` matrices should correspond with each other.

`'poly_n'`	`'kriging_all'`
`'svr_all'`	`'rbf'`
`'mars'`	`'ann'`
`'bary'`	

Table 5.1: Possible values for the metamodel parameter

The argument `method` specifies the metamodels that will be fitted. This can be one or multiple methods, so that `MF_Fit` can be run once for fitting multiple metamodels. All the metamodels that are presented earlier can be fitted. The argument `method` must be a string, or a cell of strings for multiple metamodels. An overview of strings that are accepted as a value for `method` are given in table 5.1.

`'c'`	Constant
`'l'`	Linear
`'q'`	Quadratic

Table 5.2: Available values for indicating the regression function of the kriging model

As can be seen several wildcards can be used. For example, `'poly_n'` will fit all possible polynomial models. Fitting particular polynomial models can also be done. This can be done by replacing the `'n'` in `'poly_n'` in the desired order of the polynomial model. For example, a second order polynomial model can be fitted by providing `'poly_2'` as an argument.

`'E'`	Exponential
`'EG'`	Exponential/Gaussian
`'G'`	Gaussian
`'L'`	Linear
`'S'`	Spherical
`'CC'`	Cubic
`'C'`	Cubic Spline

Table 5.3: Available values for indicating the correlation function of the kriging model

Also, the string `'kriging_all'` represents all kriging models. However, particular kriging models can also be fitted. The syntax of a kriging model is for example `'krigingcE'`, where `c` denotes the regression function and `E` denotes the correlation function. Possible options for the regression function are given in table 5.2.

An overview of possible correlation functions is given in table 5.3, where the actual corresponding correlation function can be looked up in table 3.1.

If `'svr_all'` is passed on as (one of the) argument(s) all SVR metamodels will be fitted. The SVR metamodels are distinguished by distincted kernels. The syntax of a single SVR model is given by for example`'svr_linear'`, which implies a linear kernel. The SVR metamodels that are available are given in table 5.4.

`'svr_linear'`	Linear
`'svr_poly'`	Polynomial
`'svr_rbf'`	Gaussian
`'svr_erbf'`	Exponential
`'svr_sigmoid'`	Sigmoid
`'svr_fourier'`	Fourier
`'svr_spline'`	Spline
`'svr_bspline'`	B-Spline
`'svr_anovaspline1'`	ANOVA: Spline1
`'svr_anovaspline2'`	ANOVA: Spline2
`'svr_anovaspline3'`	ANOVA: Spline3
`'svr_anovabspline'`	ANOVA: B-Spline

Table 5.4: Available kernels for SVR metamodels

A last option for the parameter `method` is the argument `'all'`. Passing this argument will cause `'MF_Fit'` to fit all possible metamodels.

The argument `param` can take several parameters which are used for both the particular metamodels and for the functionality of `MF_Fit`. The progress of `MF_Fit` can be followed, i.e. `MF_Fit` will give feedback to the user during the execution of `MF_Fit`. This can be regulated by the parameter `param.display`, which can be set to `true` (default) or `false`. Also, a waitbar can be showed for following the progress of `MF_Fit`. This is set by `param.wbar`, which also takes the boolean values. Default this is set to `false`, so no waitbar will be shown.

Other possible parameters will be explained later in Chapter 5.1.2 and Appendix A. An overview of the parameters that are discussed here and their default values is given in table 5.5.

Parameter name	Default value
`param.display`	`true`
`param.wbar`	`false`
`param.errdialog`	`false`
`param.normal`	`1`
`param.data_x`	`[]`
`param.data_y`	`[]`

Table 5.5: Overview of parameters

The output of `MF_Fit` is the variable `models`. This is a MATLAB struct containing all fitted metamodels for each response variable. The different metamodels are on the rows and the columns represent the response variables.

Each model exists of three fields, namely

- `models.model`, where the actual model and its coefficients are stored,
- `models.normalisation`, that contains the normalisation coefficients, and
- `models.name`, the name of the model is stored here.

The form of `models.model` depends on the actual metamodel and will be different for each metamodel.

5.1.2 Normalisation

In `MF_Fit` the data can also be normalised. Sometimes this is desirable for getting better and more adequate metamodel fits. Three normalisation methods are implemented in `MF_Fit`, which can be set by `param.normal`. The three normalisation methods are given by

1. No normalisation. The dataset-values will not be altered.
2. Standard normalisation. The values will be subtracted by the mean (μ) of each column of the dataset and divided by its standard deviation (σ). To each column the following formula will be applied

$$x = \frac{x - \mu}{\sigma} \tag{5.1}$$

3. $[-1, 1]$-normalisation. The values will be set in the interval $[-1, 1]$, so that the highest value is 1 and the lowest is -1. Each column will be

altered by the following formula

$$x = \frac{2x - x_{\max} - x_{\min}}{x_{\max} - x_{\min}} \tag{5.2}$$

The choice for a normalisation method can be given by setting the parameter `param.normal` to 1, 2 or 3, corresponding to the list given above.

An additional dataset for determining the normalisation values can be given by `param.data_x` and `param.data_y`. Calculation of for example the mean will then be done based on these additional datasets and with these values the original dataset will be altered. This can be useful for example when a dataset is split into a training and a validation set, and the normalisation needs to be done based on the whole dataset.

5.2 Prediction of response values

Another MATLAB function that is part of MultiFit is `MF_Evaluate`. This function can predict response values of known and unknown data points. All metamodels built with `MF_Fit` can be evaluated with this function. Given metamodel(s) and a set of input data points, `MF_Evaluate` will evaluate the provided metamodels at the data points.

5.2.1 Usage of the function

The syntax of the `MF_Evaluate` function can be described as:

```
Yvals = MF_Evaluate(Xval,models,param)
```

`MF_Evaluate` will predict the response variable(s) using the metamodel specified in `models`. Multiple metamodels can be evaluated due to the cell structure of `models`, as mentioned before with `MF_Fit`. `MF_Evaluate` will produce the predictions in a same way as `MF_Fit` produces the models, i.e. a cell array with in the rows the different metamodels and in the columns the dimensions of the explanatory variables.

Not much parameters can be passed to `MF_Evaluate`. This is because all the information about the metamodels is stored in the structure `models`. However, `param.display` and `param.wbar` can still be set to `true` or `false` for providing feedback about the progress of the function.

The output of `MF_Evaluate` is `Yvals`. This is a MATLAB struct containing the predictions of the considered data points. The rows represents the different metamodels and the columns the response variables.

5.3 Comparing accuracy of metamodels

`MF_Validate` is a covering function that makes use of `MF_Fit` and `MF_Evaluate`. It can be seen as the overall validation function. Several validation techniques can be used, such as leave-one-out and p-fold cross-validation. Local validations can be done by providing indices, so that the validation will only be based on the specified points. An optional dataset can also be passed on as a 'classical' validation.

5.3.1 Usage of the function

The syntax of the `MF_Evaluate` function can be described as:

```
validations = MF_Validate(xTrain, yTrain, fitMethod, valMethod,
                          indices, xVal, yVal,param)
```

`xTrain` and `yTrain` are similar to `Xfit` and `Yfit` and are as described so in `MF_Fit`, the actual dataset split up in response variables y and explanatory variables x. Also the variable `fitMethod`, which represents the metamodels to be validated, has the same syntax as variable `method` as described `MF_Fit`.

A new variable is `valMethod`, this variable represents the validation method. This variable takes a similar form as `fitMethod`, consisting of (a cell of) string(s), but the content is different. The variable `valMethod` represents which validation methods will be used.

`valMethod` can exists of one of more strings, each representing a validation method. The typical form of a string is for example `'loo_rmse`. As cazn be seen the string consists of two parts divided by an underscore. The first part represents the validation method and the second part the error metric. The first part can take the values described in table 5.6 and the second part the values in table 5.7

`'overall'`	Overall/residual fit
`'pfold'`	p-fold cross-validation
`'loo'`	Leave-one-out cross-validation
`'val'`	Validation based on validation set

Table 5.6: Available validation methods

If p-fold cross validation is desired, the value of p shall be given. This can be done by actually naming the value of p in the string, like `'5fold'` or `'2fold'`. When the value of p is not given, i.e. `'pfold'`, by default a 10-fold and 2-fold cross-validation will be done.

`'rmse'`	Root Mean Square Error
`'mse'`	Mean Square Error
`'rsquare'`	R-Square
`'aae'`	Average Absolute Error
`'raae'`	Relative Average Absolute Error
`'mae'`	Maximum Absolute Error
`'rmae'`	Relative Maximum Absolute Error
`'mape'`	Mean Absolute Percentage Error

Table 5.7: Available error metrics

All possible combination of the two parts are possible. Multiple validation methods can be given in a cell of strings, as explained for `method` with `MF_Fit`. Also several wildcards are available:

- `'all'`, all possible combinations of error metrics and validation methods below,
- `'overall_all'`, all metrics on the overall fit,
- `'loo_all'`, all metrics on the leave-one-out cross-validation fit,
- `'pfold_all'`, all metrics on the 2-fold and 10-fold cross-validation fit.

Chapter 6

Verification tests

To further investigate the functionality of the MultiFit toolbox, several test are done. The goal of the chapter is to verify the implementation of the metamodels, show additional features of the metamodels and give options for additional validation techniques.

6.1 Fitting polynomial models on a polynomial function

To test and assess the quality of the implemented `poly_n()` function, different two-dimensional polynomials of orders 0-12 are evaluated. The coefficients of the polynomial test functions are generated randomly. 100 data points are then generated randomly on the interval $[-10, 10]$ and the corresponding response values are calculated for the test functions. MultiFit is then used to fit these datasets with polynomial models. The considered polynomials are plotted in figure 6.1, together with the log of the RMSE's of the different fitted polynomial models.

As noted, only the polynomials of order 2-4 are shown. The other plots illustrate the same idea and are not shown here, but in Appendix C. As expected the polynomial model of the corresponding order accurately approximates the real polynomial. The polynomial of higher order than the considered order are also approximating the real polynomial really well. Polynomial models of higher order make the unnecessary coefficients zero and are thus equal to the polynomial model of the corresponding order. However, due to rounding errors, the RMSE will increase slowly with higher polynomial models. The `poly_n()` is operating as desired.

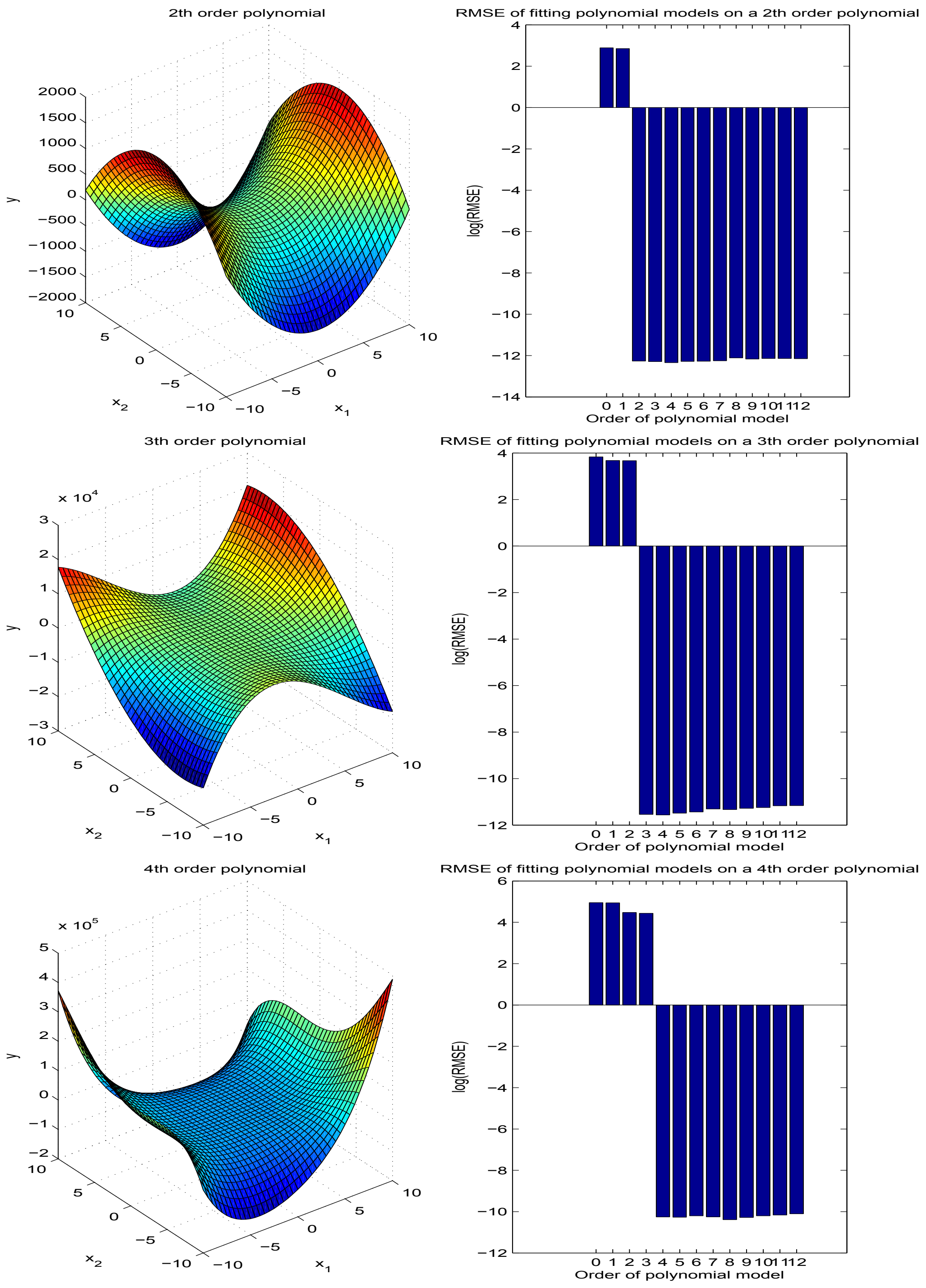


Figure 6.1: Polynomial function and metamodels

6.2 Estimation of RMSE by Kriging Models

The kriging approximation model considered in the MATLAB Kriging Toolbox DACE can make predictions of the response values at unknown design points. At known trial points the kriging model will interpolate and give a prediction equal to the observed value. This prediction is exact, i.e. there is no variance in this prediction. However, for unknown trial points the DACE predictor gives a predicted response value and a estimated mean square error (MSE). So the function returns a prediction for the response value of the unknown trial points and it makes an estimation of the distance between the prediction and the response value.

Consider the MATLAB peaks function, see equation A.11, at the interval $[-3, 3]^2$ represented by 101 data points. The leave-1-out cross-validation is done to get 101 predictions and estimated MSE's at unknown trial points. The real MSE is then calculated, by comparing the known response value with the predicted value. Both MSE's are then rescaled to RMSE, which better understandable, because it is in the same units as the observed values. First an overview of the estimated RMSE's with the real RMSE's is shown in figure 6.2.

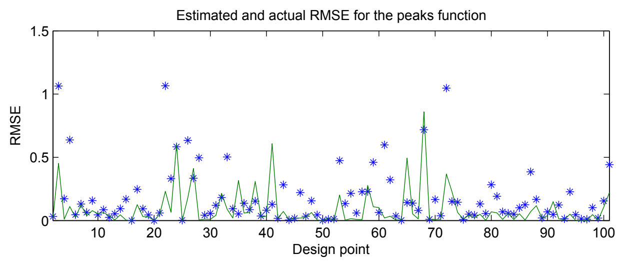


Figure 6.2: Real RMSE and the predicted RMSE by the DACE toolbox for a dataset of the Peaks test function

The stars denote the estimated RMSE, whereas the line denotes the actual RMSE. It's difficult to say something about the differences. As can be seen the estimations lie more or less around the actual values, with some outliers if prediction starts getting difficult. More can be said about the fact that the actual response value should lie in the estimated RMSE-interval around the predicted value. This is the case when the actual RMSE is less or equal to the estimated RMSE. This experiment is shown in figure 6.3.

The point below the zero-line imply that the actual response value lies in the estimate RMSE-interval. For the peaks function 76 values are in the estimated RMSE-interval.

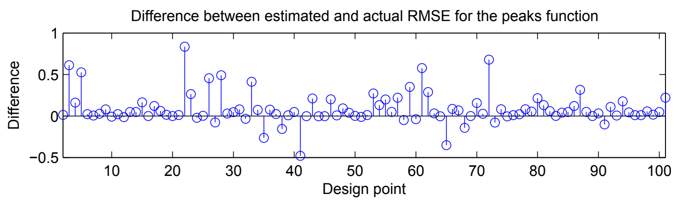


Figure 6.3: Comparison of the estimated and the real RMSE

Looking at the distribution of both the estimated and the actual RMSE's for the two datasets, shown in figure 6.4, it can be said that the estimation of the RMSE's of the wing dataset more or less covers the actual RMSE. The estimation of the RMSE of the peaks function doesn't cover the actual RMSE really well. Concluding, the estimation RMSE's can be a good indication for

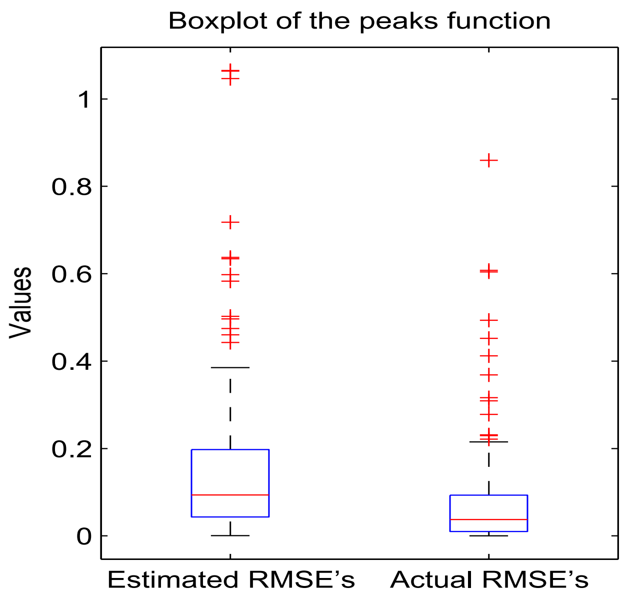


Figure 6.4: Distribution of the estimated and real RMSE's

the actual RMSE, but there is too much uncertainty to rely on the estimation. The difference between the actual and estimated value is very variable. The average of multiple estimated RMSE's can be a good approximation.

6.3 Leave-one-out cross-validation verification

Typically, the quality of a metamodel is measured using leave-one-out cross-validation. This validation method is used when no additional dataset is available. To make maximal use of the available dataset, the available dataset is used both for training and validation.

Therefore for several metamodels, the dataset is fit using leave-one-out cross validation and the obtained error is compared to the real error. This real error is calculated by an additional dataset of 1000 datapoints, a large number to ensure an approximation of the 'real' error. The dataset used for fitting the metamodels exists of 100 datapoints and multiple datasets are used, each representing one of the 14 test functions, see [36]. The error is here calculated by the RMSE. The polynomial metamodels are not fitted

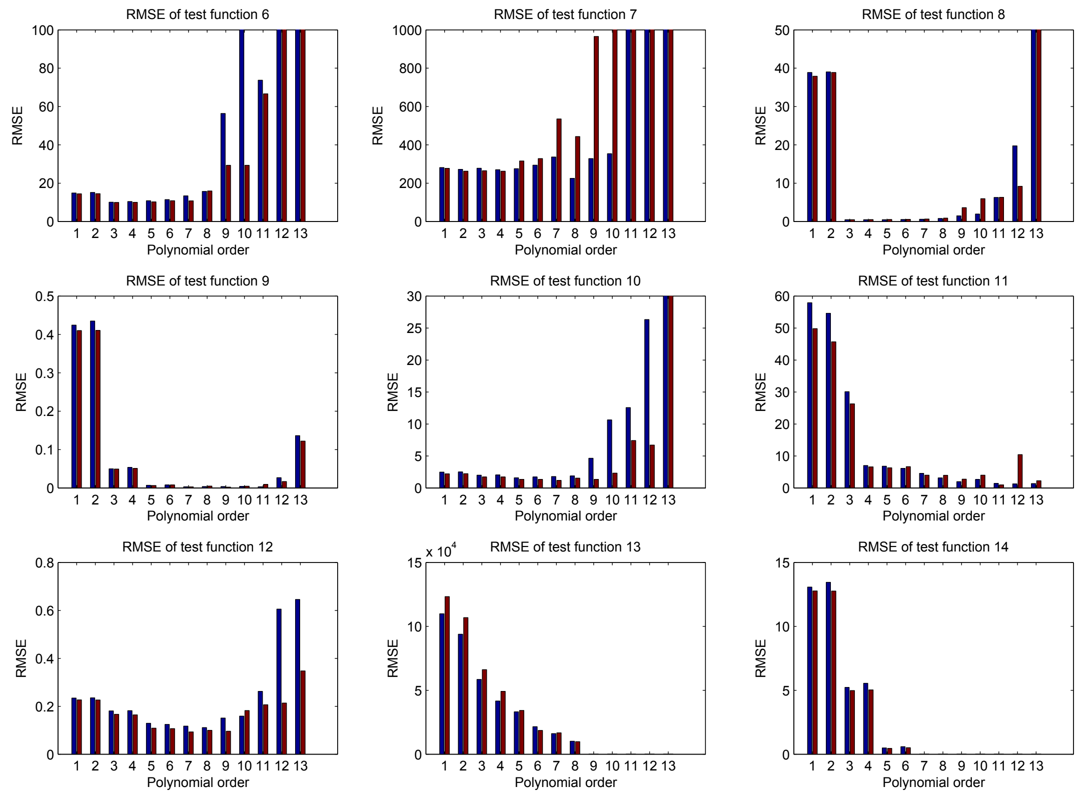


Figure 6.5: Comparison between real RMSE and RMSE estimated by leave-one-out cross-validation for polynomial models

with the first 5 test functions, because these test functions are too linear. The polynomial metamodels can approximate these test functions very accurate, and therefore are not included in this test.

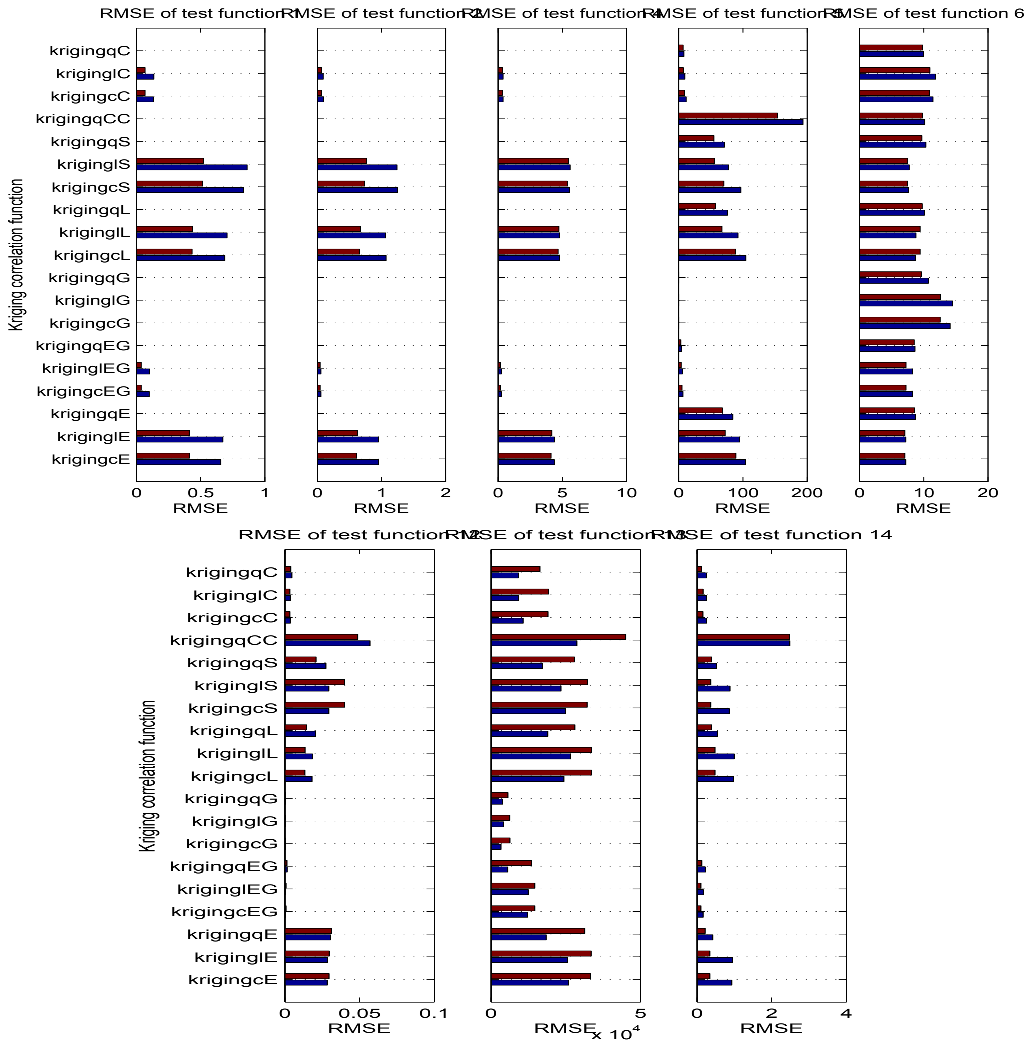


Figure 6.6: Comparison between real RMSE and RMSE estimated by leave-one-out cross-validation for kriging models

As can be seen in figure 6.5 the leave-one-out cross-validation RMSE (blue) differs not much from the real validation RMSE (red) for polynomial models. However, this is not valid for high-order polynomials of orders higher than 8. With these polynomial metamodels the two mentioned errors have a great difference. Therefore, for polynomial models of high order (9 and higher) the leave-one-out cross-validation error is not a good reflection of the real error. The error of lower-order polynomial models can be approximated by a leave-one-out cross-validation error. Also, not much can be said, whether this is an over- or underestimation.

The leave-one-out cross-validation error seem to overestimate the real validation error for kriging metamodels, as seen in figure 6.6. Note that not all test functions are shown. This is only not valid for test function 13 where leave-one-out cross-validation error is an obvious underestimation. The conclusion can be made that leave-one-out cross-validation error is not a good reflection of the real error. This is also confirmed by Meckesheimer [31], where it is concluded that leave-one-out cross-validation is not effictive for kriging models, leave-$0.1N$-out cross-validation is advised, where N is the number of datapoints.

6.4 Local validation based on Pareto ranking

The quality of a metamodel is usually measured on the whole input space. A prediction error in uninteresting sections of the input space is accounted for as much as a prediction error in an interesting error. In optimisation, it is desirable that a metamodel approximates the dataset especially around the optima well. Validation metrics usually take in account all the residuals, therefore a metric is desired, which gives more weight to the interesting areas of the input space.

Interesting sections in multi-objective optimisation can be appointed by the Pareto ranking. Pareto optimality is already discussed in Section 2.2. Pareto ranking is the ranking of all datapoints in a dataset according the theory of Pareto optimality. The Pareto optimal set of a dataset has by definition a Pareto ranking of 1. This means that these points are the optimal points in the multi-objective optimisation. The next step in appointing the Pareto ranking to the other points, is removing this Pareto optimal set from the dataset and calculating the Pareto optimal set of the reduced dataset. The points in this second Pareto optimal set get the Pareto rank 2. This algorithm continues until each point in the dataset has a Pareto ranking. At the end, each datapoint in the dataset has a Pareto ranking of 1 or higher.

Desirably a metamodel has a good local approximation of the region around points with a low Pareto ranking. If the metamodel is accurate in that area, optimisation can be better performed. This is so because optima usually occurs in the regions around or nearby the points with a low Pareto ranking. When the metamodel has a high local accuracy in the specified region, possible optimal points found during optimisation can be better approximated. A good local approximation can rule out or confirm possible optimal points.

However, optima can also occur in other regions than in the already known region of the Pareto optimal points. So note that a globally good approximation can also be preferred over a locally good approximation.

An approximation that is accurate in a specific region, but globally inaccurate, can introduce false optima. Therefore, local validation should not be the only reference for metamodel quality. Both global and local accuracy should be taken in account.

Still has to be defined what makes a region interesting. Clearly, datapoints with a low Pareto ranking are an interesting region to approximate accurate. The question arises to what rank a Pareto ranking is considered low. Taking only a validation region based on Pareto ranked 1 points can deliver a limited region and taking too much Pareto rankings can yield too many validation datapoints, causing a too large validation region to be locally significant. The number of datapoints that is locally validated must

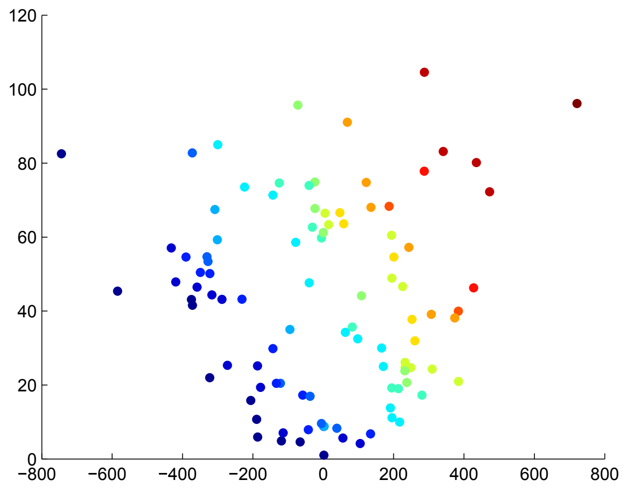


Figure 6.7: Scatter plot of the considered dataset, each data point coloured relating to its Pareto ranking

also be taken in account. A validation based on a couple of datapoints is not reliable, multiple datapoints are preferred for validation. This number of validation datapoints coheres with the maximum Pareto ranking taken in validation. So a maximum number of validation points or a maximum Pareto ranking must be decided.

Obviously, the maximum number of validation points, that is locally justified, depends on the number of datapoints in the dataset.

Two single-objective test functions, test functions number 6 and 7 as described in [36], are combined into one multi-objective test functions, which is considered to be a maximization problem. The resulting dataset is shown in figure 6.7.

Different metamodels are now fitted at this multi-objective test function. The RMSE is calculated over all datapoints, and over only Pareto ranked 1 datapoints. This set is concatenated with higher Pareto ranked points and over this set is the RMSE also calculated. In table 6.1, the overall RMSE, the RMSE over the Pareto rank 1 points, the Pareto rank 1 to 2 points, consecutively until the Pareto rank 1 to 5 points, are shown of y_1 of the most accurate metamodels of the test functions are shown.

Metamodel name	Pareto 1 RMSE	Pareto 1-2 RMSE	Pareto 1-3 RMSE	Pareto 1-4 RMSE	Pareto 1-5 RMSE	Overall RMSE
svr_rbf	357.95	319.88	297.52	282.69	277.48	257.20
rbf	1291.7	964.42	837.05	1611.2	1542.8	2910.5
mars	337.18	286.67	269.95	276.67	265.27	252.58
krigingcE	141.08	126.91	117.61	136.80	132.45	122.87
kriginglE	197.84	192.42	170.54	177.28	170.53	167.33
krigingqE	211.22	223.57	198.34	201.88	195.00	194.49
poly_1	345.79	299.41	287.46	283.37	271.95	245.8
poly_3	309.02	272.03	261.49	277.2	265.12	248.98
poly_7	289.09	257.17	234.10	277.9	266.62	282.05

Table 6.1: Results of the Pareto-based validations

As can be seen, it differs over which datapoints the metamodel is validated. Whether all datapoints or a specified set are included in the validation can make a huge difference. Therefore, there must always be taken into account whether a local or a global validation is desired. A good metamodel is accurate in both global and local areas.

Chapter 7

Case Study: Wing Data

As a case study a real aeronautic design problem is considered. This problem considers the design of a transonic wing, such that the performance of the wing is optimized. Several design properties, such as span, sweep and others, must be decided on to obtain a optimal performance values for range and fuel efficiency. This is a multi-objective optimisation problem, because both the range and the fuel efficiency must be optimized. All methods and techniques discussed herefore will be applied to this case study (where possible) . The goal is to find an accurate metamodel and optimal values for the design properties.

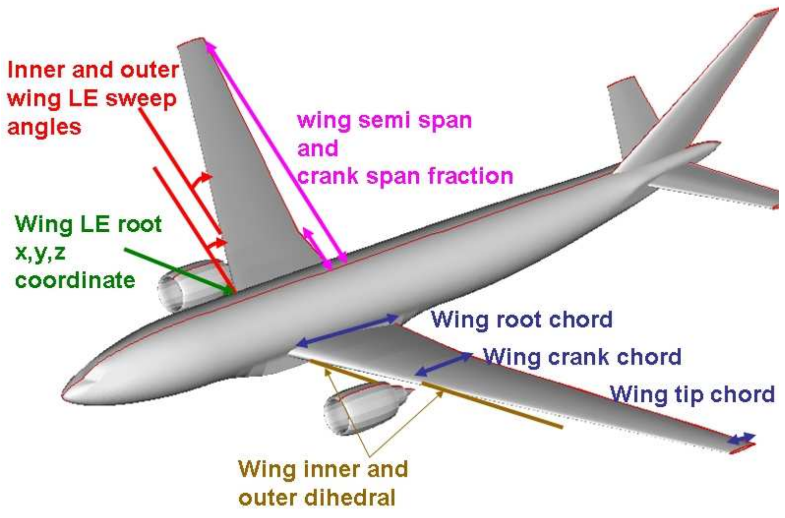

Inner and outer wing LE sweep angles
wing semi span and crank span fraction
Wing LE root x,y,z coordinate
Wing root chord
Wing crank chord
Wing tip chord
Wing inner and outer dihedral

Figure 7.1: Wing geometrics and parameters

7.1 The dataset

The dataset that is used in this case study for wing design, consists of 4 independent explanatory variables and 2 observation values. The 4 explanatory variables stand for wing semi-span in meters, outer-wing-leading-edges sweep angle in degrees, relative chord change, and maximum takeoff weight in 10^5 kilograms. The explanatory variables are respectively denoted as x_1, x_2, x_3 and x_4. The span, sweep and chord are illustrated in Figure 7.1. The MTOW represents the maximum total mass of the aircraft at take-off.

The response values that must be optimized are the fuel efficiency in km/(l/pax) and the Breguet range in nautical miles. These are denoted as y_1 and y_2. Note that an 'absolute' optimal value may not exist, therefore a Pareto optimal set is searched. Also note that this is considered to be a maximization problem.

A dataset consisting of 99 datapoints is available for this design problem. So 99 different design points with corresponding response values are available. To explore the dataset, both response values are shown in a scatter plot in Figure 7.2. Each datapoint is coloured corresponding to its Pareto rank as obtained for the range and fuel-efficiency objectives. Points with a low Pareto rank are blue coloured, whereas points with a high Pareto rank are red coloured.

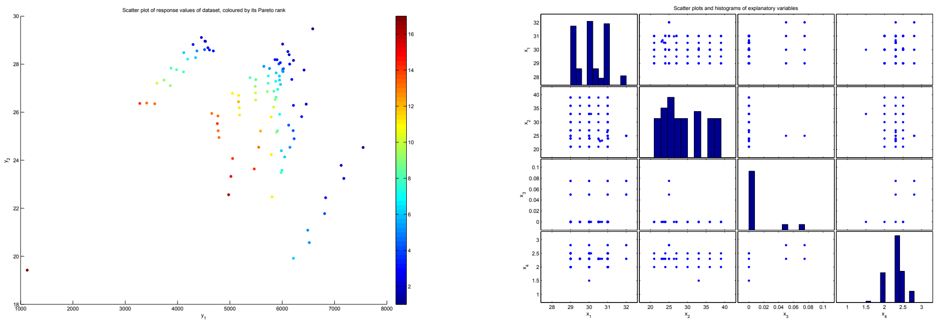


Figure 7.2: Exploration of dataset

The characteristics of the explanatory variables of the dataset are also shown conveniently arranged. A scatter plot is made of each possible combination of explanatory variables, and a histogram is shown of each explanatory variable.

An outlier is obviously visible in Figure 7.2, the point in the lower left corner of the scatter plot of the response values, which correlates with the point with the lowest value of x_4. This datapoint is clearly numerically distant from the rest of the datapoints. Also, the region of this datapoint is uninteresting, because the optimisation is a maximization problem. The metamodel to be build should be accurate in regions interesting for maximization. Including this datapoint in the dataset, the metamodel will consider the outlier and the uninteresting region, while it is desirable to have an accurate metamodel in interesting regions. Therefore, this datapoint is left out of the considered dataset. This yields a dataset of 98 points, and the scatterplot of the response values is given in Figure 7.3.

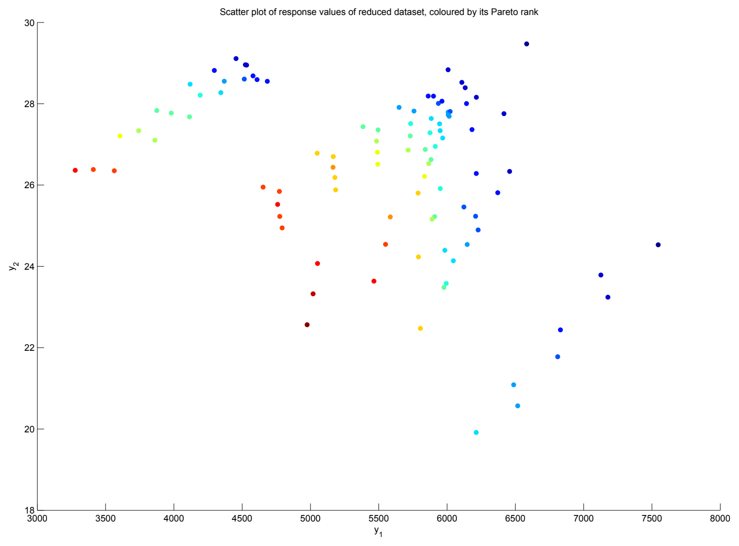


Figure 7.3: Reduced dataset

In Figure 7.4 multiple scatter plots are made of the explanatory variables against the response variables. Like the scatter plot of the response values, each datapoint is coloured corresponding its Pareto rank.

Points with a high value of x_3 have predominantly low Pareto rankings, which is seen by the blue colour of the datapoints. High values of x_3 seem to be correlated with the Pareto optimal set. Optimal points seem also to be located around a value of 25 for x_2. A trend is obviously visible for x_4 for both response values. A positive trend is visible for y_1 and a negative trend for y_2.

These observations can also be described in words. A high relative chord change and a sweep of around 25 degrees seems to be optimal. And, as expected, an high take-off weight will decrease the range but increase the fuel efficiency and vice versa for a low take-off weight.

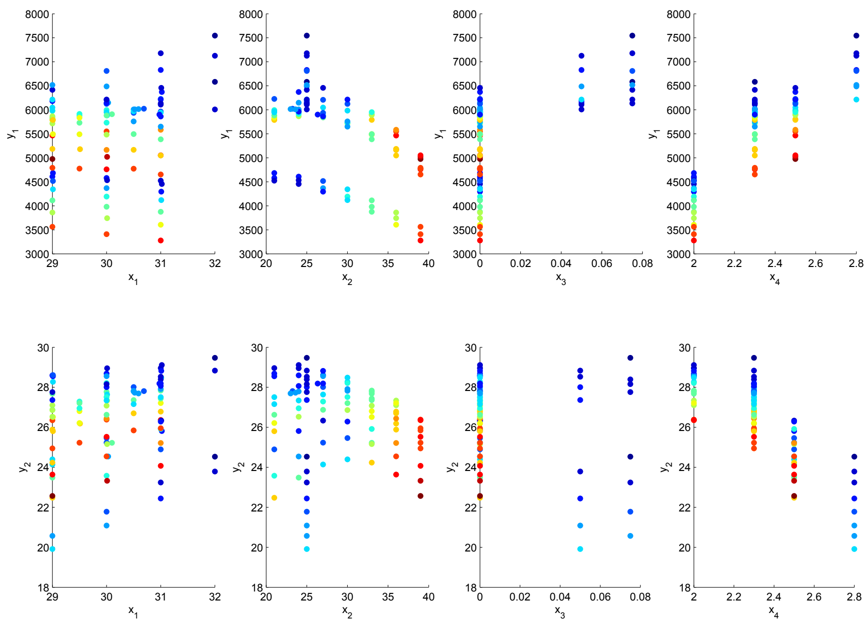


Figure 7.4: Relation between explanatory and reponse variables

7.1.1 Normalisation

Calculating with very large or small values can cause numerical problems. These numerical problems can occur when dealing with a dataset with values around zero and large values. Therefore a normalisation of the dataset can be desirable in that case. Rank deficiency problems in polynomial models, as explained in Section 3.1, can also be reduced by normalizing the dataset.

The two normalization methods discussed in that Section are applied to this dataset. This is done because this dataset contains zero values, which can cause numerical problems. Two datasets are therefore now obtained. One dataset is centered to the mean of each explanatory variable and one dataset is transformed on the interval $[-1, 1]$.

For illustration, a boxplot of the original and the two normalized values of x_3 is shown in Figure 7.5. As can be seen with normalization method 2, the values of x_3 are all located in the interval $[-1, 1]$. The centering of normalization method 3 is not visible, because the boxplot only displays the median and not the mean, what would be zero. Notable is that the distribution of the datapoints is still retained throughout the normalization methods.

The case study will be continued with the normalized datasets. However, for simplicity reasons only one dataset will be described, namely the centered dataset. Throughout the test will emerge which normalisation method will be chosen eventually.

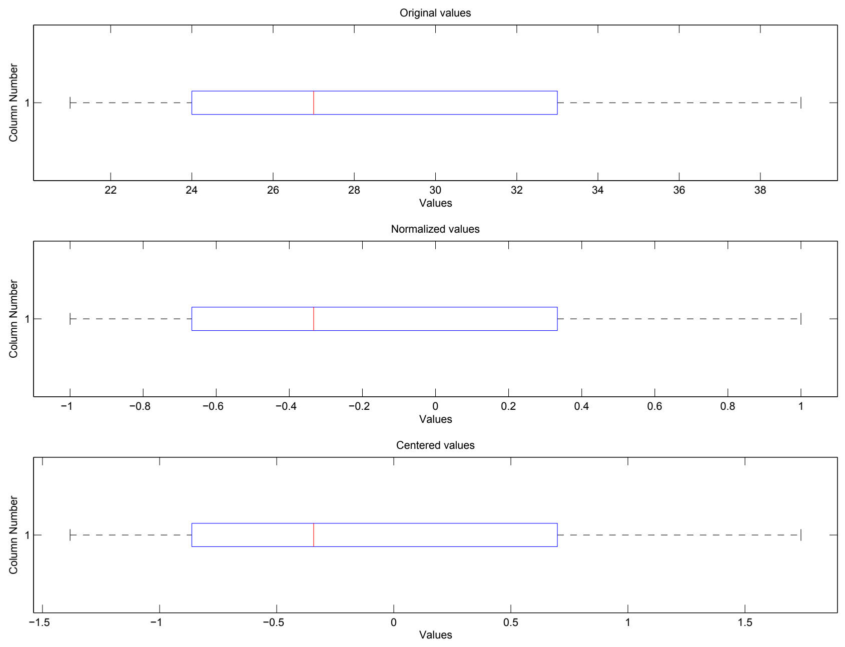


Figure 7.5: Boxplots of normalization methods

7.2 Metamodel fitting

The dataset is examined and made ready for fitting metamodels. A metamodel is single-objective, while this problem is multi-objective, so two metamodels will be build.

Due to the abundance of metamodels which will be difficult to conveniently arrange, the metamodels will be considered in methodical order. A preselection will be made of the methods: polynomial, kriging and svr models. One metamodel will be chosen from each method to make an empirical comparison with the other methods. First off are the polynomial models.

7.2.1 Polynomial models

Polynomial models of all possible orders are now fitted on the normalized datasets. For this dataset the maximal order of a polynomial model is 4, polynomial models of higher order give an underdetermined system. The polynomial metamodels are fitted on the whole dataset and the residuals are taken from the predictions of the original datapoints.

All considered metrics indicate poly_3 as the most accurate metamodel for the dataset. This is a polynomial metamodel of order 3. This cubic polynomial model achieves the lowest value for each metric. Both response values seem to agree about the favoured metamodel.

Metamodel name	RMSE	R-Square	AAE	MAE	MAPE
y_1					
poly_0	885.62	0.00000	724.75	2209.5	0.14565
poly_1	335.6	0.8564	278.96	953.97	0.052705
poly_2	76.312	0.99258	57.643	229.03	0.011001
poly_3	40.024	0.99796	28.597	153.22	0.0052697
poly_4	85.769	0.99062	52.47	438.91	0.0097188
y_2					
poly_0	2.0245	0.00000	1.6067	6.4476	0.063594
poly_1	0.91983	0.79356	0.77063	2.4264	0.029811
poly_2	0.18015	0.99208	0.14015	0.4432	0.0053603
poly_3	0.075774	0.9986	0.058683	0.28294	0.0022008
poly_4	0.20867	0.98938	0.14259	0.7667	0.0055478

Table 7.1: Error metrics of polynomial models based on error of approximation

Note that rank deficiency occurs with polynomial models of order 2 and higher. This causes that small alterations in the explanatory variables can cause great differences in the predictions. This can be seen by the differences in the error metrics of the two datasets with the polynomial models of order 2 and higher. Whereas the lower-order polynomial models doesn't differentiate, the higher-order polynomial models has some variance between the two datasets.

The predictions given by models where rank deficiency occurred are not reliable. Therefore, the `poly_1` metamodel is selected. The performance of the second order polynomial model is worse, but rank deficiency will not occur here.

7.2.2 Kriging models

Now the kriging models will be fitted to select one or multiple metamodels for the comparison study. Leave-one-out cross-validation is applied to the original dataset, because kriging metamodels are interpolating metamodels and residuals are thus zero. The leave-one-out predictions are then measured in the error metrics denoted in Table 7.2.

Metamodel name	RMSE	R-Square	AAE	MAE	MAPE
y_1					
krigingcE	92.931	0.98899	34.801	510.36	0.005699
kriginglE	86.276	0.99051	31.301	510.86	0.0051875
krigingcEG	88.243	0.99007	35.999	550.4	0.0061304
kriginglEG	81.853	0.99146	32.575	533.53	0.0056309
krigingcG	86.644	0.99043	51.389	442.81	0.009454
kriginglG	67.322	0.99422	37.965	359.53	0.0070063
krigingcL	109.85	0.98461	43.438	517.12	0.0070402
kriginglL	105.25	0.98588	41.32	513.12	0.0067478
krigingcS	92.994	0.98897	38.135	475.88	0.0061841
kriginglS	106.79	0.98546	42.685	511.42	0.0069714
krigingcCC	582.13	0.56794	378.42	2037.4	0.071113
kriginglCC	258.44	0.91484	165.93	1000.6	0.030892
krigingcC	79.773	0.99189	36.787	456.89	0.0062757
kriginglC	77.24	0.99239	38.095	449.65	0.0065491
y_2					
krigingcE	0.16445	0.9934	0.078917	0.77835	0.0030034
kriginglE	0.15341	0.99426	0.080144	0.75616	0.0030331
krigingcEG	0.18387	0.99175	0.11194	0.90974	0.0042336
kriginglEG	0.17056	0.9929	0.096774	0.8347	0.0036158
krigingcG	0.21954	0.98824	0.13538	0.87085	0.005445
kriginglG	0.26235	0.98321	0.18105	0.94658	0.0070297
krigingcL	0.14069	0.99517	0.062895	0.79035	0.0023567
kriginglL	0.1611	0.99367	0.086156	0.76202	0.0032555
krigingcS	0.21449	0.98878	0.085147	1.2246	0.0032691
kriginglS	0.15175	0.99438	0.076583	0.77042	0.0028707
krigingcCC	1.5488	0.41472	0.9166	6.2374	0.037482
kriginglCC	0.64353	0.89896	0.4604	2.1557	0.018235
krigingcC	0.17571	0.99247	0.099894	0.8427	0.0037588
kriginglC	0.16797	0.99312	0.10356	0.74801	0.0039183

Table 7.2: Error metrics of global validation of anisotropickriging models

Note that kriging models with quadratic regression functions are not fitted. This is due to the rank deficiency of these models for the considered dataset. The kriging metamodels are then not reliable and thus ignored.

For the first response variable y_1, the `kriginglG` and `kriginglC` metamodels perform good on both the RMSE and the R-Square error metrics. However, on the error metrics AAE and MAPE, `kriginglE` and `kriginglEG` have the best performance.

The second response variable y_2 is obviously represented best by the `krigingcL` metamodel. This metamodel scores best on all error metrics, except the MAE. Second up, is the `krigingcL` metamodel. Other two metamodels that score good, are the both metamodels with an exponential correlation function, `krigingcE` and `kriginglE`.

However, also kriging models with isotropic parameters are available. These isotropic metamodels are also fitted and the results are shown in Table 7.3.

Metamodel name	RMSE	R-Square	AAE	MAE	MAPE
y_1					
krigingcE	104.31	0.98613	44.71	466.63	0.0072415
kriginglE	96.128	0.98822	41.96	451.09	0.006857
krigingcEG	71.541	0.99347	31.27	437.41	0.0052549
kriginglEG	68.214	0.99407	30.865	402.05	0.005154
krigingcG	159.72	0.96747	80.264	899.35	0.013845
kriginglG	109.83	0.98462	63.156	456.64	0.011372
krigingcL	104.64	0.98604	44.766	472.02	0.0072497
kriginglL	97.178	0.98796	42.112	455.85	0.0068627
krigingcS	105.3	0.98586	44.691	471.9	0.0072303
kriginglS	95.858	0.98828	41.307	452.55	0.0067446
krigingcCC	590.85	0.5549	395.17	2032.9	0.077245
kriginglCC	243.64	0.92432	161.26	935.09	0.029876
krigingcC	74.226	0.99298	34.279	421.2	0.0058105
kriginglC	68.535	0.99401	33.51	392.09	0.0056678
y_2					
krigingcE	0.19882	0.99036	0.10555	0.76419	0.0040708
kriginglE	0.17646	0.9924	0.098733	0.72922	0.0037873
krigingcEG	0.1717	0.99281	0.1048	0.96142	0.0039968
kriginglEG	0.1442	0.99493	0.091446	0.55184	0.0034924
krigingcG	0.78455	0.84982	0.42493	3.7491	0.017362
kriginglG	0.40951	0.95908	0.27138	1.5004	0.01075
krigingcL	0.19818	0.99042	0.10578	0.7754	0.004076
kriginglL	0.17653	0.9924	0.099346	0.73696	0.0038077
krigingcS	0.20231	0.99001	0.10684	0.77873	0.0041189
kriginglS	0.17734	0.99233	0.10013	0.71854	0.0038381
krigingcCC	1.5741	0.39543	0.94419	6.2691	0.038358
kriginglCC	0.64471	0.89859	0.43963	2.18	0.017464
krigingcC	0.16924	0.99301	0.095155	0.94798	0.0036412
kriginglC	0.14301	0.99501	0.085058	0.57099	0.003251

Table 7.3: Error metrics of global validation of isotropic kriging models

Isotropic kriging models with the Gaussian and the cubic spline correlation functions perform bad on both response values. These 4 metamodels achieve the worst values for all error metrics for both response values. The metamodels that seem to perform the best are the 4 metamodels with the correlation functions, Exponential/Guassian and Cubic.

The isotropic and anisoptrics kriging models selected before are now mutual compared and shown in Table 7.4.

Metamodel name	RMSE	R-Square	AAE	MAE	MAPE
y_1					
kriginglE	86.276	0.99051	31.301	510.86	0.0051875
kriginglEG	81.853	0.99146	32.575	533.53	0.0056309
kriginglG	67.322	0.99422	37.965	359.53	0.0070063
kriginglC	77.24	0.99239	38.095	449.65	0.0065491
krigingcEG (iso)	71.541	0.99347	31.27	437.41	0.0052549
kriginglEG (iso)	68.214	0.99407	30.865	402.05	0.005154
krigingcC (iso)	74.226	0.99298	34.279	421.2	0.0058105
kriginglC (iso)	68.535	0.99401	33.51	392.09	0.0056678
y_2					
krigingcE	0.16445	0.9934	0.078917	0.77835	0.0030034
kriginglE	0.15341	0.99426	0.080144	0.75616	0.0030331
krigingcL	0.14069	0.99517	0.062895	0.79035	0.0023567
kriginglS	0.15175	0.99438	0.076583	0.77042	0.0028707
krigingcEG (iso)	0.1717	0.99281	0.1048	0.96142	0.0039968
kriginglEG (iso)	0.1442	0.99493	0.091446	0.55184	0.0034924
krigingcC (iso)	0.16924	0.99301	0.095155	0.94798	0.0036412
kriginglC (iso)	0.14301	0.99501	0.085058	0.57099	0.003251

Table 7.4: Error metrics of selected isotropic and anisotropic kriging models

Obviously, the isotropic `krigingcL` is chosen for the second response variable. This metamodel scores best on all error metric except MAE. However, this error metric only represents an error in one point, instead of an overview of all errors. So, that is considered to be an exception.

For the first response variable, the choice for a metamodel is more difficcult. The `kriginglG` performs best on RMSE, R-Square and MAE, but the isotropic `kriginglEG` metmamodel scores best for AAE and MAPE. Because MAPE and RMSE are not correlated and both represent a different picture, both metamodels are selected for future study.

7.2.3 SVR models

There are a lot of support vector regression metamodels available. Each one has an own kernel. All these metamodels are fitted on the dataset and the error metrics of the leave-one-out residuals are calculated and shown in Table 7.5. Leave-one-out cross-validation is applied here, because the RBF and the exponential RBF kernels are interpolating. Also, the kernels spline, sigmoid, anovaspline1, anovaspline2 and anovaspline3 are not tested here, because these kernels yield predictions for this dataset, that are unreasonably high.

Metamodel name	RMSE	R-Square	AAE	MAE	MAPE
y_1					
svr_linear	360.29	0.8345	296.38	1054.0	0.055924
svr_poly	361.68	0.83322	167.57	1689.4	0.029488
svr_rbf	826.87	0.12827	644.5	2135.8	0.13008
svr_erbf	883.51	0.0047719	712.1	2225.2	0.14361
svr_fourier	894.75	-0.020725	732.22	2232.3	0.14716
svr_bspline	341.98	0.85089	180.74	1593.0	0.032903
svr_anovabspline	167.95	0.96403	85.49	873.43	0.01551
y_2					
svr_linear	0.98154	0.76493	0.81653	2.6297	0.031629
svr_poly	0.79425	0.84608	0.33381	4.2008	0.012455
svr_rbf	1.9139	0.1063	1.412	6.514	0.056418
svr_erbf	2.0113	0.012985	1.5674	6.514	0.062202
svr_fourier	2.0453	-0.020725	1.6232	6.5141	0.06425
svr_bspline	0.99103	0.76037	0.49115	4.7382	0.020609
svr_anovabspline	0.59733	0.91294	0.29134	2.9713	0.012134

Table 7.5: Error metrics of selected SVR models

The SVR models seem not be able to represent the dataset adequately. The only SVR metamodels that somehow approximate the dataset are the ones with a anova b-spline kernel, `svr_anovabspline`, which will be selected in the future study.

7.2.4 Final comparison of metamodels

The preselection of metamodels is done and the selected metamodel for further examination are listed here.

- `poly_1`
- `kriginglG` (for y_1)
- `kriginglEG (iso)` (for y_1)
- `krigingcL (iso)` (for y_2)

- `svr_anovabspline`
- `rbf`
- `mars`
- `ann`

To assess the quality of each metamodels, several cross-validations are used. The validation techniques leave-one-out, 2-fold, and 10-fold cross-validation are used for determining the residuals. Fitting these metamodels on the whole dataset and predicting all the points will yield incomparable results, due to the interpolating character of some of these metamodels.

Three metrics are chosen to be shown, because they each show an other aspect of the fit. The MAE shows the maximum error, and thus the accuracy of the metamodel in its worst case. The RMSE shows the overall mean error, with a negative bias toward large errors, whereas the MAPE accounts for the overall mean relative error.

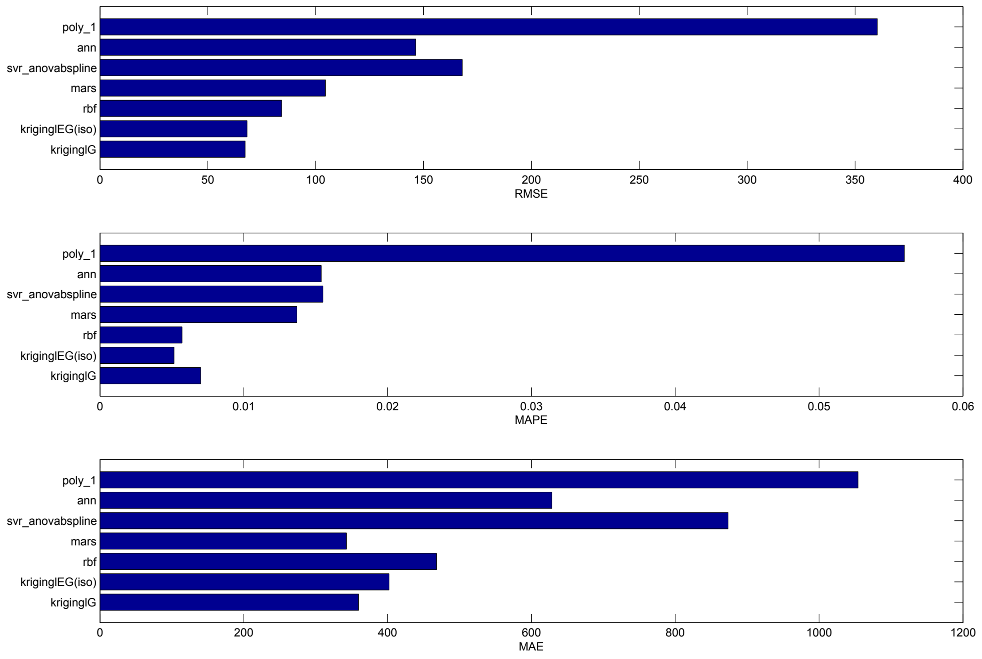


Figure 7.6: Leave-one-out cross-validation

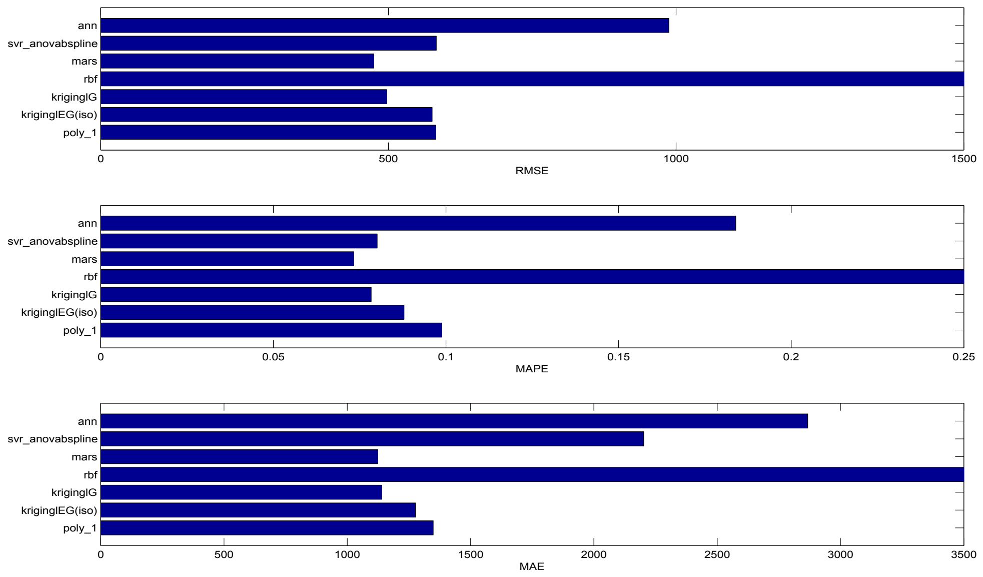


Figure 7.7: 2-fold cross-validation

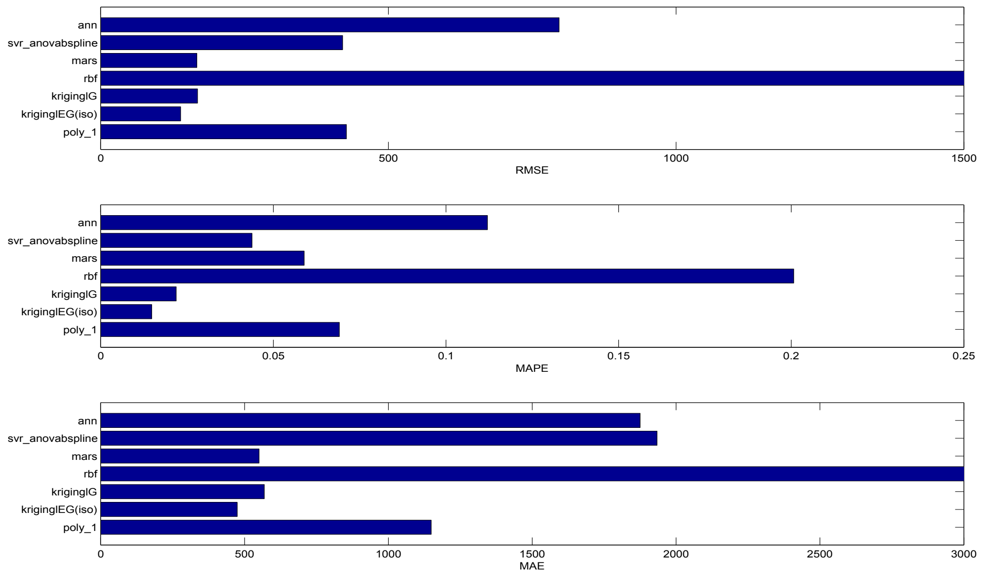


Figure 7.8: 10-fold cross-validation

These cross-validations show all a different picture. In the 2-fold cross-validation, shown in Figure 7.7, the `mars` metamodel has a very good performance. This metamodel seem to be able to make relatively accurate predictions with a small dataset. In 2-fold cross-validation the dataset is split in two, and each half is used to predict the other half. The `kriginglG` metamodel also has a very good performance on all three error metrics.

The radial basis functions metamodels perform very bad with p-fold cross-validations, although they perform reasonably good on leave-one-out cross-validation.

In the 10-fold cross-validation the `kriginglEG(iso)` metamodel achieves the lowest value in each error metric. Also, the `kriginglG` and `mars` metamodels scores high values.

The most important cross-validation is the leave-one-out cross-validation, whereas the p-fold cross-validations can give a good background picture. The both kriging metamodels don't differ much from eachother in the RMSE of the leave-one-out cross-validations. The MAPE and the p-fold cross-validations seem to be biased towards the `kriginglEG(iso)`. Therefore, this metamodel is chosen for the first response value.

The results of the second response value are now discussed.

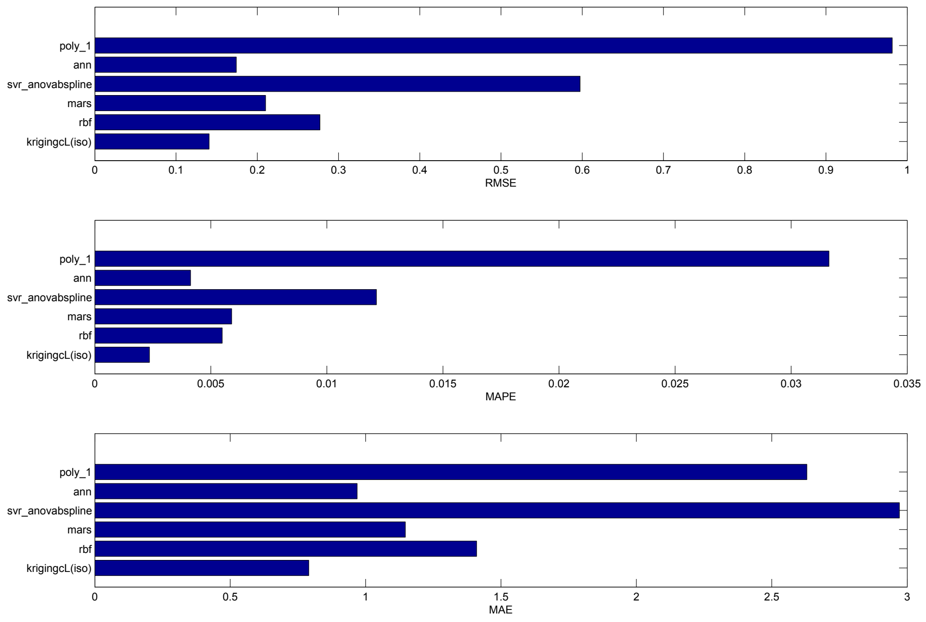


Figure 7.9: Leave-one-out cross-validation

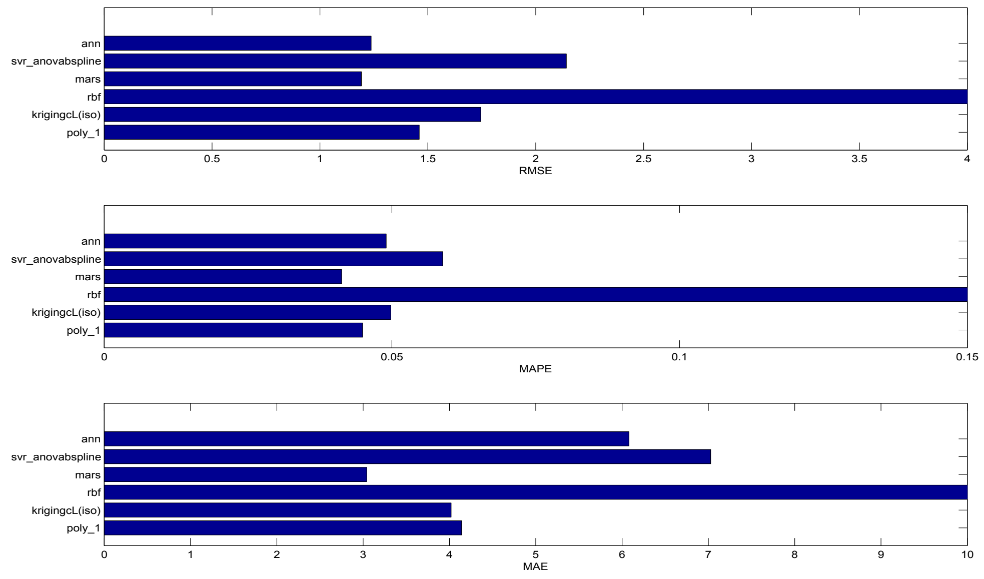


Figure 7.10: 2-fold cross-validation

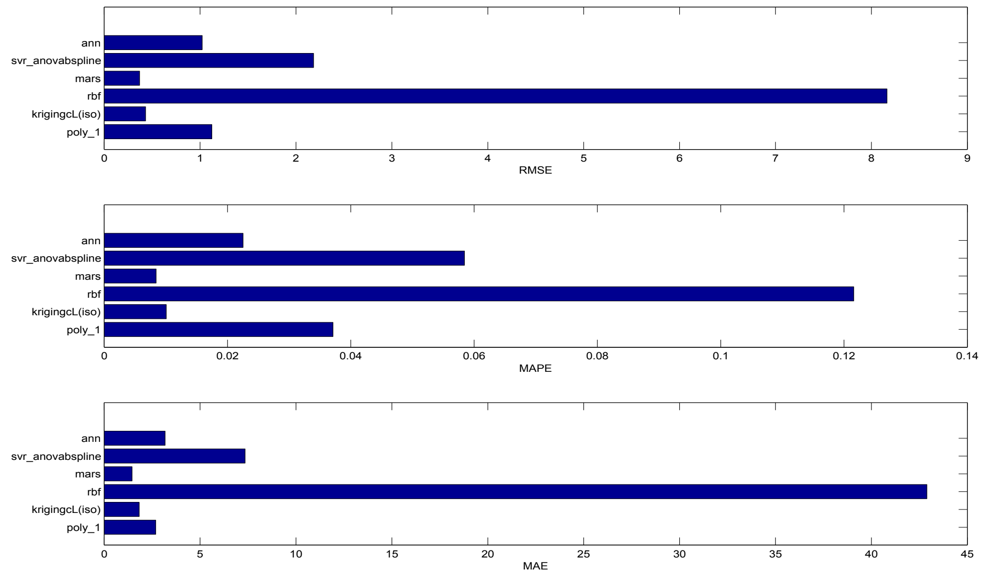


Figure 7.11: 10-fold cross-validation

Notable is that the MARS metamodel outperforms the kriging and all other metamodels on all error metrics in the *p*-fold cross-validations seen in Figures 7.11 and 7.10. However, in the leave-one-out cross validation the kriging metamodel has a significant better performance. The *p*-fold cross-validations are validations with smaller datasets, and thus less information about the dataset. A metamodel is expected to have a good performance on the whole dataset. Therefore, the `krigingcL(iso)` is chosen to model the second response variable.

7.3 Optimisation

It is concluded that the best fit for fuel efficiency is the isotropic kriging model with a linear regression function and the Exponential/Gaussian correlation function. The best fit for the range is also a isotropic kriging model, but this time with a constant regression function and a linear correlation function. Both metamodels are interpolating models, which means that the models yield exact predictions, this is also desired.

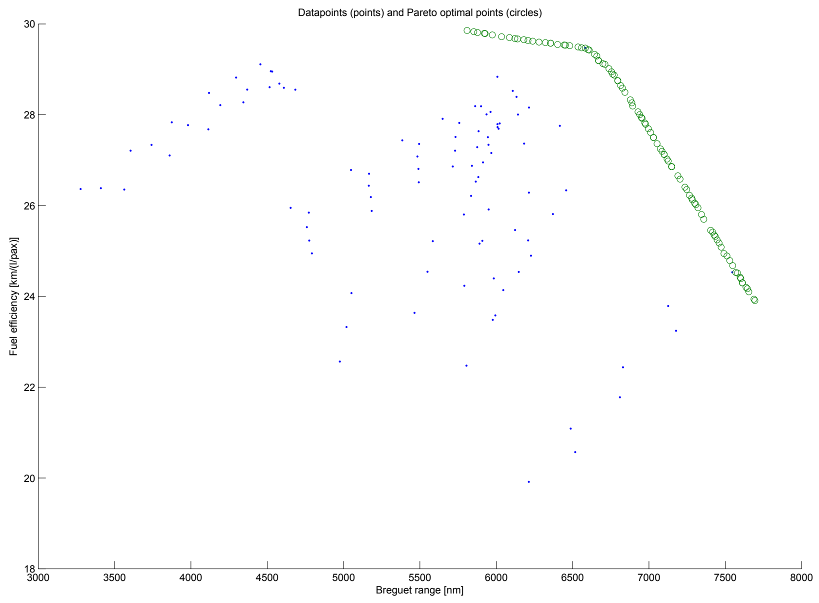


Figure 7.12: Pareto optimal set (circles) for maximum range and maximum fuel efficiency found by optimisation with metamodels

A Pareto front is now searched using these metamodels as the objective functions for the multi-objective optimisation. This is done using a genetic algorithm, based on NSGA-II as described in [20].

In this optimisation a population of 98 individuals is used, where the initial population is given by the original dataset. In the first run 9 generations are considered, which again yields 98 individuals. These individuals are then used in an extensive optimisation run of 100 generations. The upper and lower bounds for the individuals are given by the minimum and maximum values of the original dataset.

As can be seen in Figure 7.12 a Pareto front is found. The points in this front are the optimal set of datapoints found by the optimisation procedure. The optimized set of points obtain clearly better values for the objective functions, comparing to the original dataset.

As stated in Section 7.2, the kriging metamodels provide an additional value for uncertainty, the MSE. The kriging metamodels can indicate an expected error for each datapoint. Taking the square root of the provided MSE, an approximation for the expected error of approximation can be found.

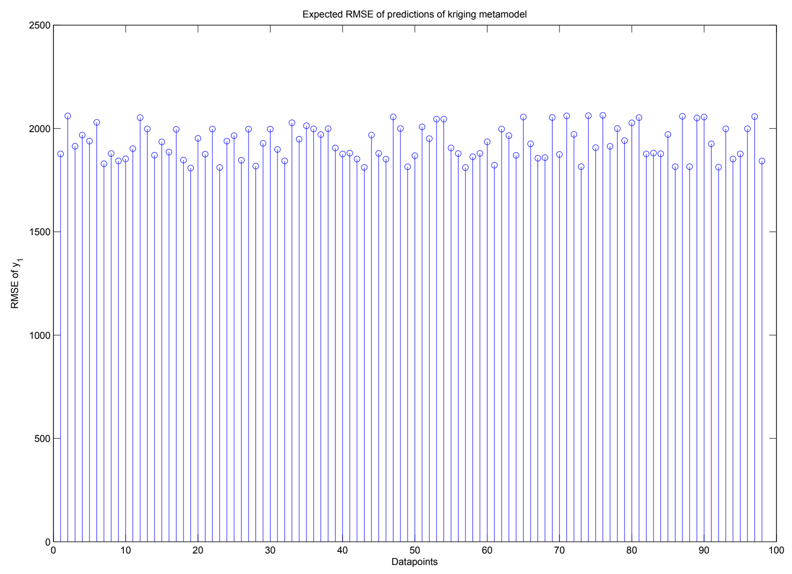


Figure 7.13: Square root of MSE's of kriging model of fuel efficiency.

The kriging metamodel for the second response variable, the kriging model with a linear correlation function, gives an equal MSE for each datapoints, due to its linearity. The square root of the MSE of the second response variable for each of the Pareto optimal points are given in Figure 7.13.

Chapter 8

Conclusion/General Discussion

Many different metamodels can be fitted on a dataset. Not only are there a lot of distinctive metamodels, each metamodel also provides different options for parameters, kernel functions, correlation functions and other settings that are part of a metamodel. Each dataset needs an individual approach for choosing a metamodel and its settings. No generic rule can be given, that is based on characteristics of a dataset. In order to choose a metamodel that adequately describes the design problem, comparison between the different metamodels and its settings are needed.

Also, the definition of a metamodel that 'adequately describes a design problem' should be considered. If a problem involves an exploration of the whole design space, the problem needs a metamodel that is globally accurate. Whereas an optimisation problem, that is known to have an optimum in a certain area of the design space, should especially be accurate in that area. The accuracy of a metamodel is linked to the information known about the design space and thus to the problem.

Not only should be questioned in which area the metamodel should be accurate, also the significance of an eventual residual should be questioned. Large residuals can be weighted more in order to find the best metamodel, i.e. with a lot of small residuals in contrast to less large residuals. However, this can cause problems with design problems with very large and small response values, because the residuals at the very large residuals will make the residuals at the small response values insignificant. In this situation, a measure relative to the response value should be chosen. So, also the representation of the error of approximation should be chosen, paying importance to the characteristics of the design problem.

To fully use the available dataset, the dataset should be used both for fitting and for validation, which is called cross-validation. Leave-one-out cross-validation seems to be utilized mostly in the literature. This cross-validation technique is also intuitively prefered. Leave-one-out cross-validation seems to take maximal information out of the dataset. However, it is questionable, because leave-one-out cross-validation never considers the full dataset.

The computational time of leave-one-out cross-validation increases sharply together with the size of the dataset. In combination with leave-one-out cross-validation computational efficient metamodels should be chosen, such as kriging or polynomial models. Kriging metamodels seem to be favoured above all the other metamodels, due to their small computational time and accuracy.

It is advised that all metamodels should be fitted, together with leave-one-out cross-validation. However, not always enough time is available. As said, kriging metamodels are generally preferred, but other metamodels still can outperform this metamodel depending on the datasets.

The results should still be compared and investigated. All error metrics and validation techniques should be taken into account and an appropriate one should be chosen for each dataset and design problem.

The different metamodels can easily be fitted, evaluated and validated with the implemented functions in MultiFit. A leave-one-out cross-validation for all metamodels can for example be done in one invocation of the implemented function.

MultiFit should be used by users with some knowledge about metamodels and validation methods. Because no generic rules are available for choosing metamodels, the results should all be taken into account together with the information about the design problem. For users with some knowledge in the field of metamodelling, the MultiFit provides a transparant, easy environment for creating and evaluating metamodels.

Bibliography

[1] J.D. Anderson. *Computational Fluid Dynamics: The Basics with Applications*. McGraw-Hill, 1995.

[2] A.J. Booker. *Global Modeling for Optimization Boeing/IBM/Rice Collaborative Project, 1995 Final Report*. Boeing Information & Support Services, 1995.

[3] A.J. Booker. Design and analysis of computer experiments. *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 7 th, St. Louis, MO*, pages 118–128, 1998.

[4] R. A. Bradley and S. S. Srivatava. Correlation in polynomial regression. *Amer. Statistician*, 33:11, 1979.

[5] B. Cheng and DM Titterington. Neural Networks: A Review from a Statistical Perspective. *Statistical Science*, 9(1):2–30, 1994.

[6] Nello Cristianini and John Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.

[7] C. Curin, T. Mitchell, M. Morris, and D. Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Amer. Statist. Assoc.*, 86:953–963, 1991.

[8] N. Dyn, D. Levin, and S. Rippa. Numerical Procedures for Surface Fitting of Scattered Data by Radial Functions. *SIAM Journal on Scientific and Statistical Computing*, 7:639, 1986.

[9] H. Fang, M. Rais-Rohani, and M.F. Horstemeyer. Multiobjective crashworthiness optimization with radial basis functions. *Proceedings of the 10 thAIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Paper No. AIAA-2004-4487, Albany, NY, AIAA*, 2004.

[10] J.H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1–67, 1991.

[11] A.A. Giunta and L.T. Watson. A comparison of approximation modeling techniques- Polynomial versus interpolating models. *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 7 th, St. Louis, MO*, pages 392–404, 1998.

[12] A.S. Goldberger. Best Linear Unbiased Prediction in the Generalized Linear Regression Model. *Journal of the American Statistical Association*, 57(298):369–375, 1962.

[13] S.R. Gunn. Support Vector Machines for Classification and Regression. *ISIS Technical Report*, 14, 1998.

[14] RL Hardy. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. Res*, 76(8):1905–1915, 1971.

[15] R. Jin, W. Chen, and TW Simpson. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization*, 23(1):1–13, 2001.

[16] PK Kitanidis. Parameter Uncertainty in Estimation of Spatial Functions: Bayesian Analysis. *Water Resources Research WRERAO Vol. 22*, (4):499–507, 1986.

[17] J.P.C. Kleijnen. *Statistical tools for simulation practitioners.* Marcel Dekker, Inc. New York, NY, USA, 1986.

[18] J.P.C. Kleijnen and R.G. Sargent. *A methodology for fitting and validating metamodels in simulation.* Center for Economic Research, Tilburg University, 1997.

[19] JR Koehler, AB Owen, S. Ghosh, and CR Rao. Computer experiments. *Handbook of Statistics*, 13:261–308, 1996.

[20] J.B. Kollat and P.M. Reed. The Value of Online Adaptive Search: A Performance Comparison of NSGAII, ε-NSGAII and εMOEA. *Evolutionary Multi-Criterion Optimization*, 2005.

[21] AN Kolmogorov. On the representation of continuous function of many variables by superposition of continuous functions of one variable and addition. *American Mathematical Society Translation*, 28:55–59, 1963.

[22] T. Krishnamurthy. Response surface approximation with augmented and compactly supported radial basis functions. *44 th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2003.

[23] G.M. Laslett. Kriging and Splines: An Empirical Comparison of Their Predictive Performance in Some Applications. *Journal of the American Statistical Association*, 89(426), 1994.

[24] Y. Lin. An Efficient Robust Concept Exploration Method and Sequential Exploratory Experimental Design. 2004.

[25] S.N. Lophaven, H.B. Nielsen, and J. Sondergaard. DACE-A Matlab Kriging Toolbox, Version 2.0. *IMM Technical University of Denmark, Lyngby*, 2002.

[26] KV Mardia and RJ Marshall. Maximum Likelihood Estimation of Models for Residual Covariance in Spatial Regression. *Biometrika*, 71(1):135–146, 1984.

[27] H.C. Martin and G.F. Carey. *Introduction to finite element analysis.* McGraw-Hill New York, 1973.

[28] J.D. Martin and T.W. Simpson. Use of Kriging Models to Approximate Deterministic Computer Models. *AIAA Journal*, 43(4):853–863, 2005.

[29] M. Meckesheimer. *A Framework For Metamodel-Based Design: Subsystem Metamodel Assessment And Implementation Issues.* PhD thesis, The Pennsylvania State University, 2001.

[30] M. Meckesheimer, R.R. Barton, T. Simpson, F. Limayem, and B. Yannou. Metamodeling of combined discrete/continuous responses. *AIAA Journal*, 39(10):1950–1959, 2001.

[31] M. Meckesheimer, A.J. Booker, R.R. Barton, and T.W. Simpson. Computationally inexpensive metamodel assessment strategies. *AIAA Journal*, 40(10):2053–2060, 2002.

[32] Toby J. Mitchell and Max D. Morris. The spatial correlation function approach to response surface estimation. In *WSC '92: Proceedings of the 24th conference on Winter simulation*, pages 565–571, New York, NY, USA, 1992. ACM.

[33] R.H. Myers and D.C. Montgomery. *Response Surface Methodology: Process and Product in Optimization Using Designed Experiments*. John Wiley & Sons, Inc. New York, NY, USA, 1995.

[34] I.G. Osio and C.H. Amon. An engineering design methodology with multistage Bayesian surrogates and optimal sampling. *Research in Engineering Design*, 8(4):189–206, 1996.

[35] V. Pareto. *Manuale di Economia Politica*. Piccola Biblioteca Scientifica, Milan, 1906. Translated into English by Ann S. Schwier (1971), Manual of Political Economy, MacMillan, London.

[36] H Pohlheim. Geatbx examples: Examples of objective functions. 2005.

[37] MJD Powell. Radial basis functions for multivariable interpolation: a review. *Clarendon Press Institute Of Mathematics And Its Applications Conference Series*, pages 143–167, 1987.

[38] D.E. Rumelhart, B. Widrow, and M.A. Lehr. The basic ideas in neural networks. *Communications of the ACM*, 37(3):87–92, 1994.

[39] J. Sacks, S.B. Schiller, and W.J. Welch. Designs for Computer Experiments. *Technometrics*, 31(1):41–47, 1989.

[40] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):409–423, 1989.

[41] M. Shacham and N. Brauner. Minimizing the Effects of Collinearity in Polynomial Regression. *Ind. Eng. Chem. Res*, 36(10):4405–4412, 1997.

[42] T.W. Simpson, T.M. Mauery, J.J. Korte, and F. Mistree. Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA Journal*, 39(12):2233–2241, 2001.

[43] TW Simpson, JD Poplinski, PN Koch, and JK Allen. Metamodels for Computer-based Engineering Design: Survey and recommendations. *Engineering with Computers*, 17(2):129–150, 2001.

[44] A.J. Smola, B. Schölkopf, and K.R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998.

[45] Alex J. Smola and Bernhard Scholkopf. On a kernel-based method for pattern recognition, regression, approximation, and operator inversion. *Algorithmica*, 22(1/2):211–231, 1998.

[46] M.L. Stein. A Comparison of Generalized Cross Validation and Modified Maximum Likelihood for Estimating the Parameters of a Stochastic Process. *The Annals of Statistics*, 18(3):1139–1157, 1990.

[47] CH Tu and RR Barton. Production yield estimation by the metamodel method with a boundaryfocused experiment design. *Design Theory and Methodology Conference–DTM97*, 1997.

[48] V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. *Advances in Neural Information Processing Systems*, 9:281–287, 1997.

[49] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24(6):774–780, 1963.

[50] V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 2000.

[51] G. Wahba. A Comparison of GCV and GML for Choosing the Smoothing Parameter in the Generalized Spline Smoothing Problem. *The Annals of Statistics*, 13(4):1378–1402, 1985.

[52] G.G. Wang and S. Shan. Review of Metamodeling Techniques in Support of Engineering Design Optimization. 2006.

[53] W.J. Welch, R.J. Buck, J. Sacks, H.P. Wynn, T.J. Mitchell, and M.D. Morris. Screening, Predicting, and Computer Experiments. *Technometrics*, 34(1):15–25, 1992.

[54] H. Wendland. On the smoothness of positive definite and radial functions. *Journal of Computational and Applied Mathematics*, 101(1):177–188, 1999.

[55] C.H. Wu, J.M. Ho, and D.T. Lee. Travel-time prediction with support vector regression. 5(4):276–281, December 2004.

[56] Z. Wu. Multivariate compactly supported positive definite radial functions. *Adv. Comput. Math*, 4:283–292, 1995.

[57] RJ Yang, N. Wang, CH Tho, JP Bobineau, and BP Wang. Metamodeling Development for Vehicle Frontal Impact Simulation. *Journal of Mechanical Design*, 127:1014, 2001.

Appendix A

Tests for determining default settings

The implementation of the metamodels in the MultiFit-environment is described in Chapter 5. For verification of the new software implementation, many tests have to be done in order to verify the possible settings of the functions. In the following sections, each metamodel will be tested for the different input settings.

The test functions considered in this chapter are taken from [36], unless mentioned otherwise.

A.1 Kriging metamodels

The kriging metamodels are implemented in MultiFit through the DACE Matlab toolbox, which is described in [25]. All regression and correlation functions mentioned in Section 3.2 are available for fitting metamodels. One correlation function (exponential/Gaussian in Table 3.1 requires a parameter. The correlation function is given by

$$\exp\left(-\theta_k \left|\boldsymbol{x}_k^{(i)} - \boldsymbol{x}_k^{(j)}\right|^d\right), \tag{A.1}$$

where d denotes the power of the difference and this parameter is set by `param.expg`.

To determine what an appropriate value of `param.expg` would be, datasets based on 14 different test functions are fit twice, once with 100 samples and once with 50 samples. The fit is done with kriging metamodels with constant, linear and quadratic regression functions. The parameter is varied from 1.1 to 1.9, this is done because when the value of `param.expg` is equal to 1 or 2,

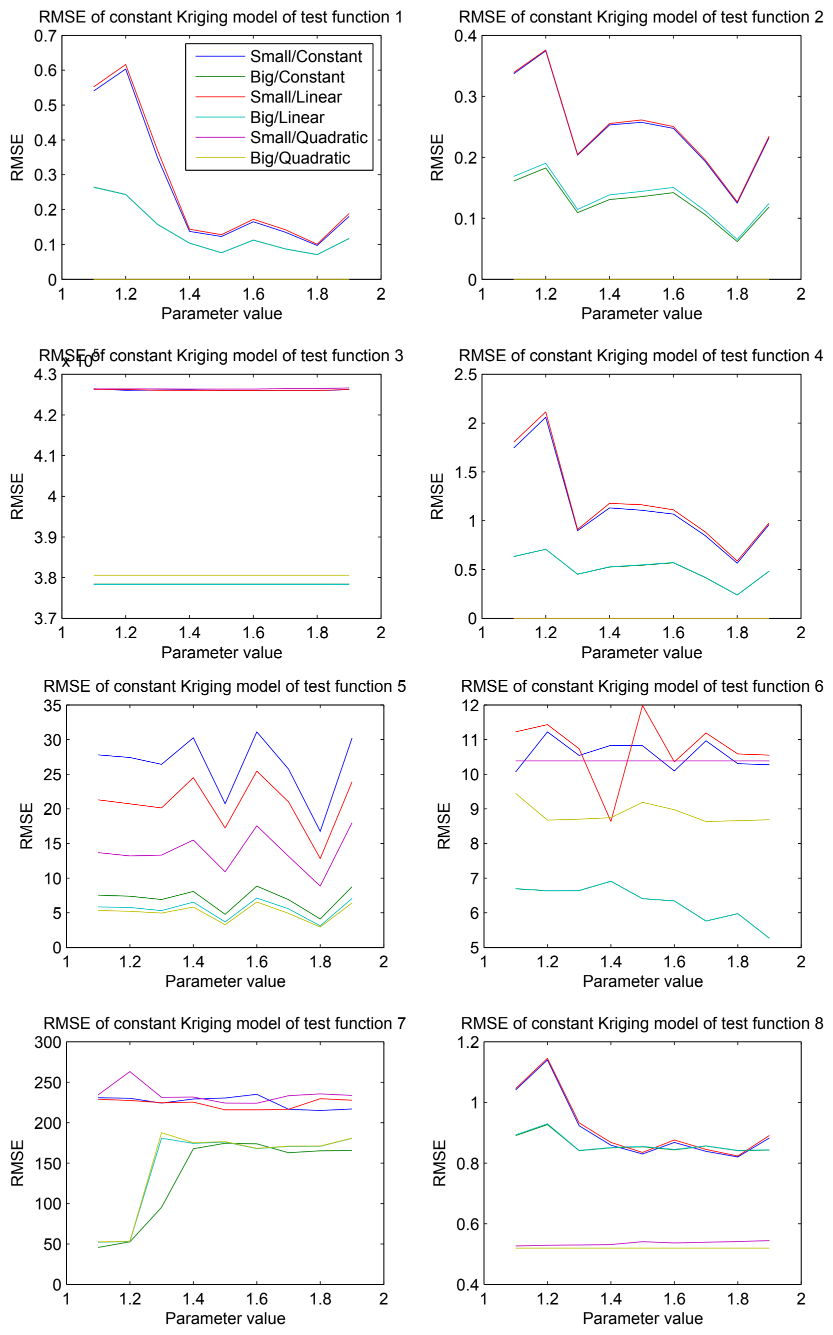


Figure A.1: Testing different values for the parameter of the Exponential/Gaussian correlation function

other already implemented correlation functions will result. The performance is then measured by comparing the predictions of the metamodels in 1000 validation points with their actual values. The RMSE of these predictions is then calculated to provide a surveyable comparison. The plots of these RMSE results are shown in figure A.1 and A.2 .

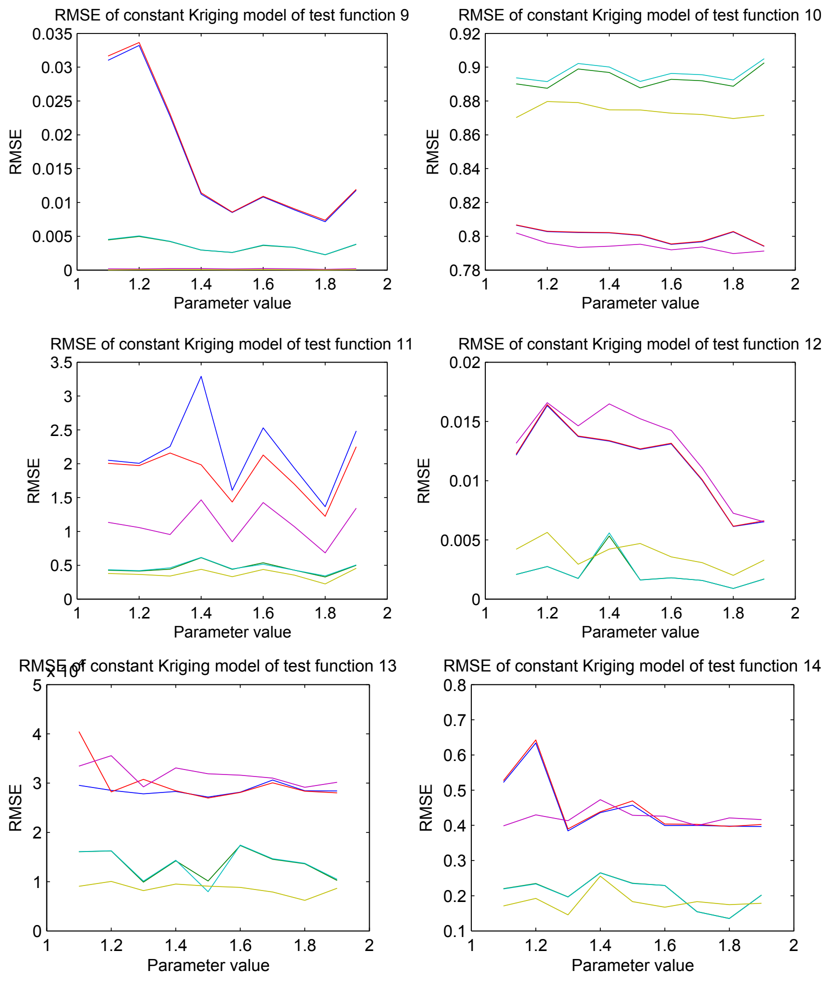


Figure A.2: Testing different values for the parameter of the Exponential/Gaussian correlation function (continued)

The legend of the plots is given in the first plot, the plot of test function 1. Most RMSE of the metamodels of the test functions seem to ascend or oscillate around a certain level. For many of the tests, the RMSE tend to go up for high values of d. The minimum RMSE of most metamodels is found at a value of d of 1.8. Therefore a default value of 1.8 is chosen for `param.expg`.

The next parameter that is tested, is the parameter that decides upon

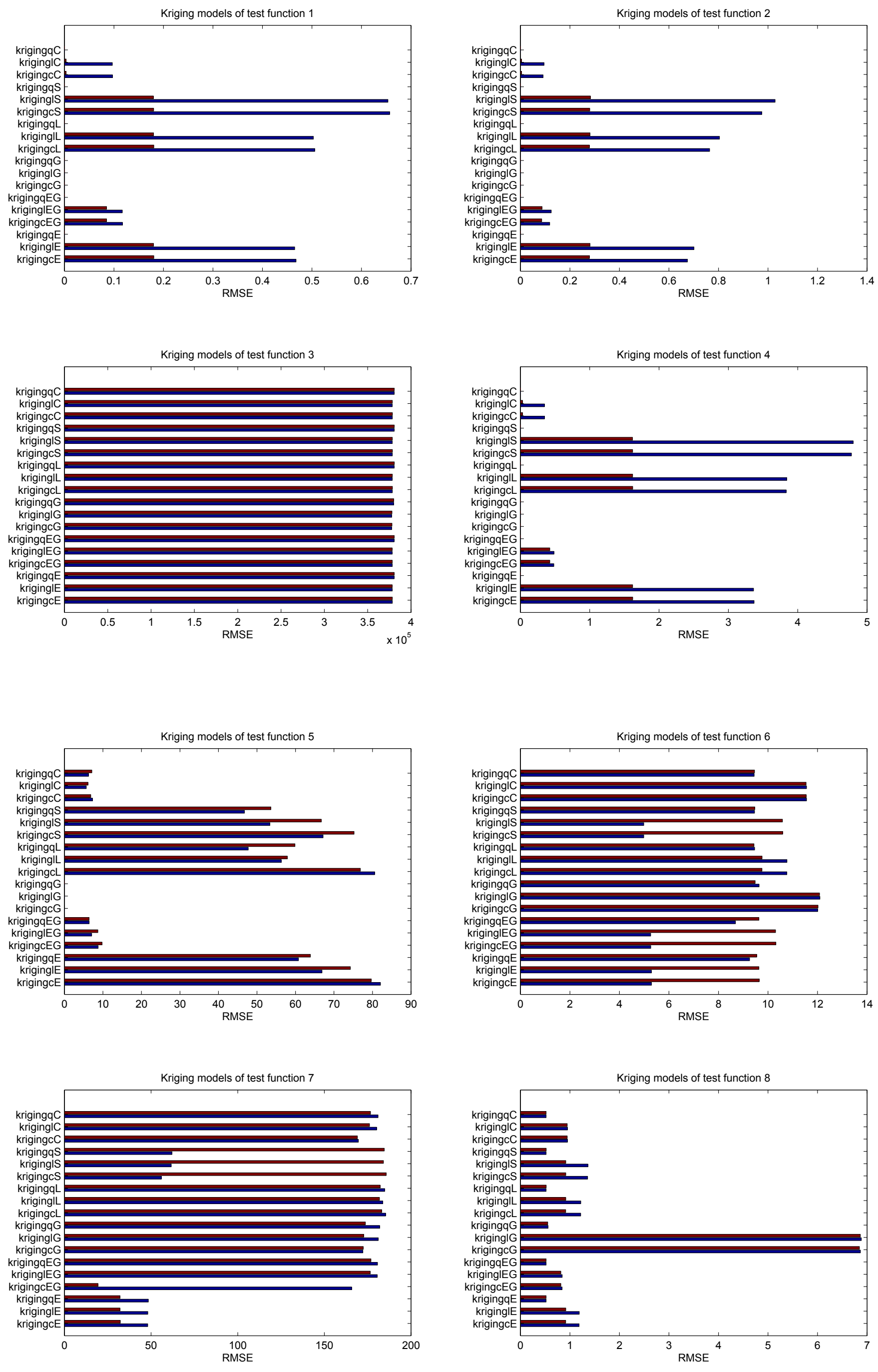


Figure A.3: Isotropic and anisotropic kriging models fit on different test functions

the isotropic character of the resulting kriging model. This parameter, given by `param.isotropic` can be set to `true` or `false`. Isotropy is described in section 3.2 and will not be further explained here. To determine the default value of `param.isotropic`, the 14 datasets based on the test functions are again fitted and the metamodels will be isotropically and anisotropically fit. All kriging metamodels are fitted, only kriging metamodels with a cubic spline correlation function are left out. This is done because this correlation function yields extremely bad results in comparison to the other correlation functions. The results are shown below.

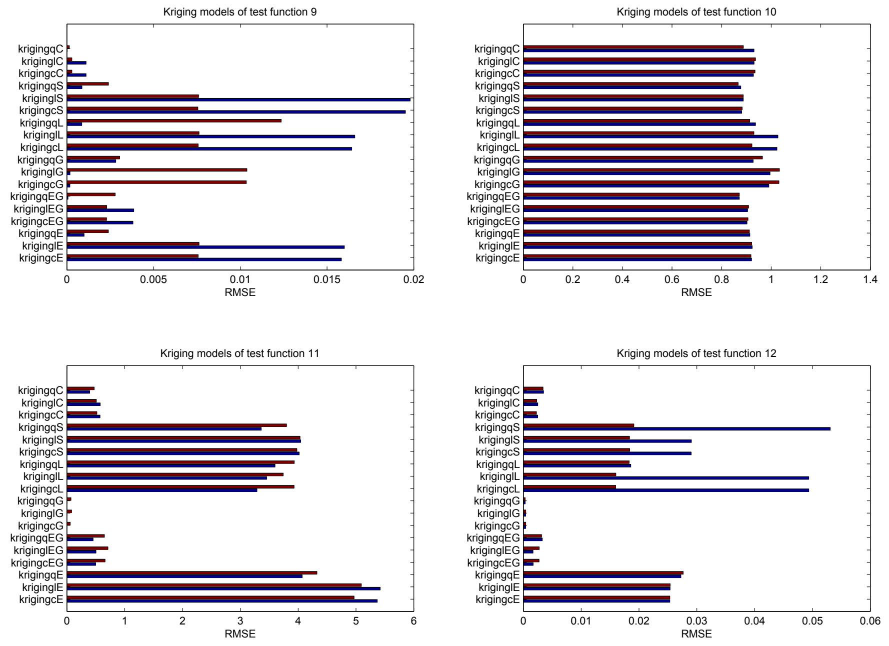


Figure A.4: Isotropic and anisotropic kriging models fit on different test functions (continued)

The red bars correspond with the isotropic kriging metamodels. First is noted that test function 3 yield indecisive results, so this test function is ignored. Looking at the plots, the functions 1,2,4,7,8 and 12 seem to favour isotropic kriging metamodels. However, the test functions 5,6 and 11 reach a lower RMSE when fitting anisotropic Kriging metamodels. Test function 9,10, 12 and 14 are indecisive and not much can be said about the difference between anisotropic and isotropic metamodels. An isotropic metamodel will be faster to compute, because less optimisations have to be done. The Kriging metamodels, however, already perform reasonably fast, so saving computational time is not the priority. Comparing anisotropic and isotropic kriging metamodels, the performance of the metamodels depends on the dataset and which correlation and regression function is used for fitting the Kriging metamodel. Since it is noted in [25] that "phenomenons are often anisotropic", the default value of `param.isotropic` is set to `false`.

A.2 RBF

Another metamodel that can be fitted with `MF_Fit` are radial basis functions metamodels. These metamodels are explained in Section 3.4. Radial basis functions are implemented in `MF_Fit` by the MATLAB-function `newrbe`, which is an interpolating radial basis function.

Only Gaussian radial basis functions will be considered here. The form of the Gaussian radial basis function is determined by its width parameter, see c in equation 3.26. This parameter is also incorporated in `MF_Fit` by the parameter `param.spread`. This parameter can be set for determining the Gaussian radial basis function.

To see what influence the parameter `param.spread` has on the performance of the metamodel, metamodels with varying width parameter are fit on the 14 test functions, as mentioned in Appendix. Two randomly generated datasets are used here, a big dataset consisting of 100 samples and a small dataset consisting of 50 samples.

The width parameter coincidences with the spread of the input space. The width parameter must be larger than the minimum distance between two adjacent input vectors, but smaller than the distance across the whole input space. This can be computed with a function called `compute_spreads`. Therefore, the width parameter of the RBF metamodels are varied between the minimum and the maximum width.

Again a validation set of 1000 data points is generated to measure the performance of the fitted metamodels. This is done by the RMSE, given in equation 4.2. The results of the 14 test functions are plotted below.

The plots are adjusted, so in some of the figures, the whole plot is not visible. There can be assumed that the part of the plot that is not shown is ascending steadily from a high value. As can be seen, the RMSE of radial basis functions metamodels with a width parameter set to the minimum value, is very high. This is thus not preferable. Also, with the last 4 test functions, test functions 11 to 14, the RMSE seems to increase when the width parameter is approximating the maximum value. At most of the test functions, the RMSE of the RBF metamodels seem to reach a low value

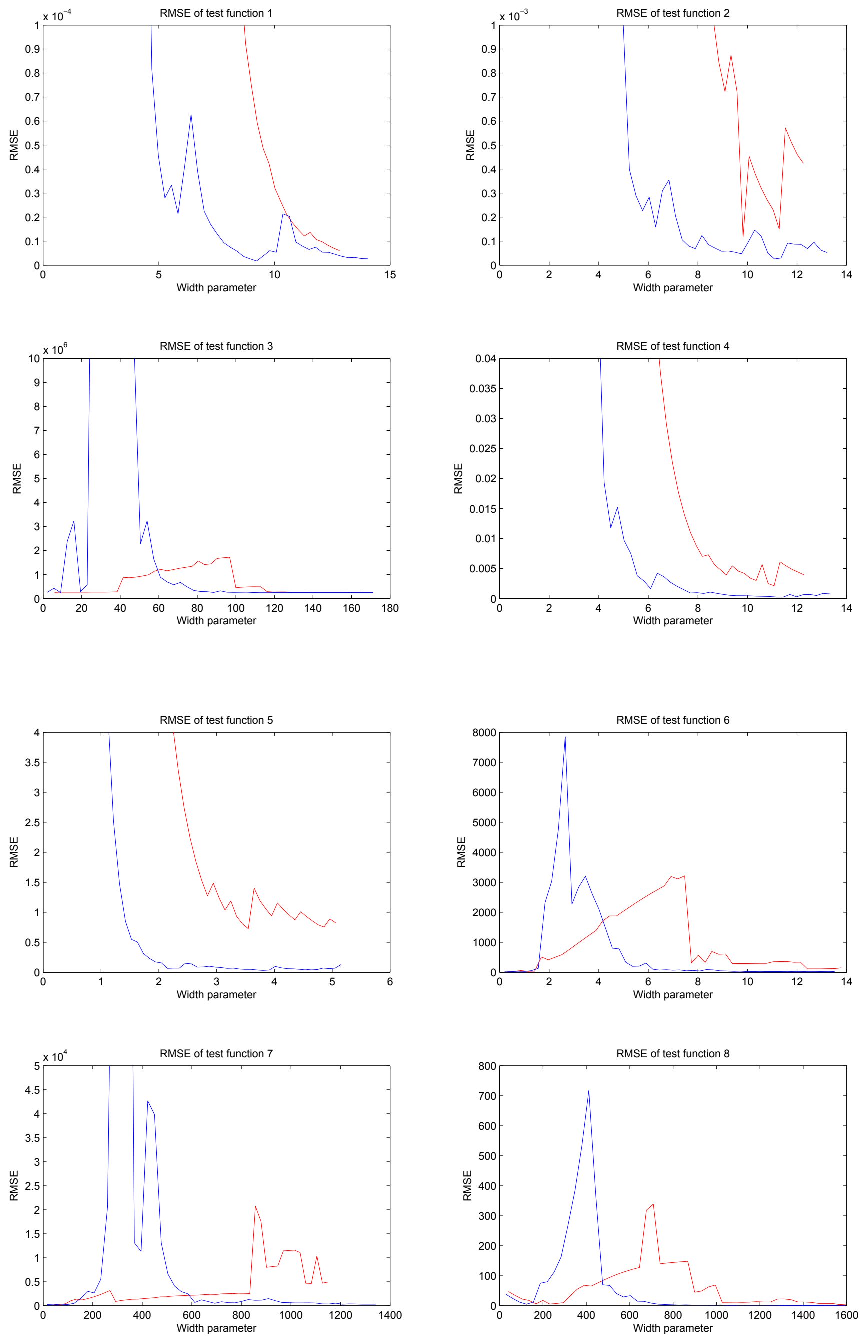


Figure A.5: Influence of the width parameter on the accuracy of the meta-model

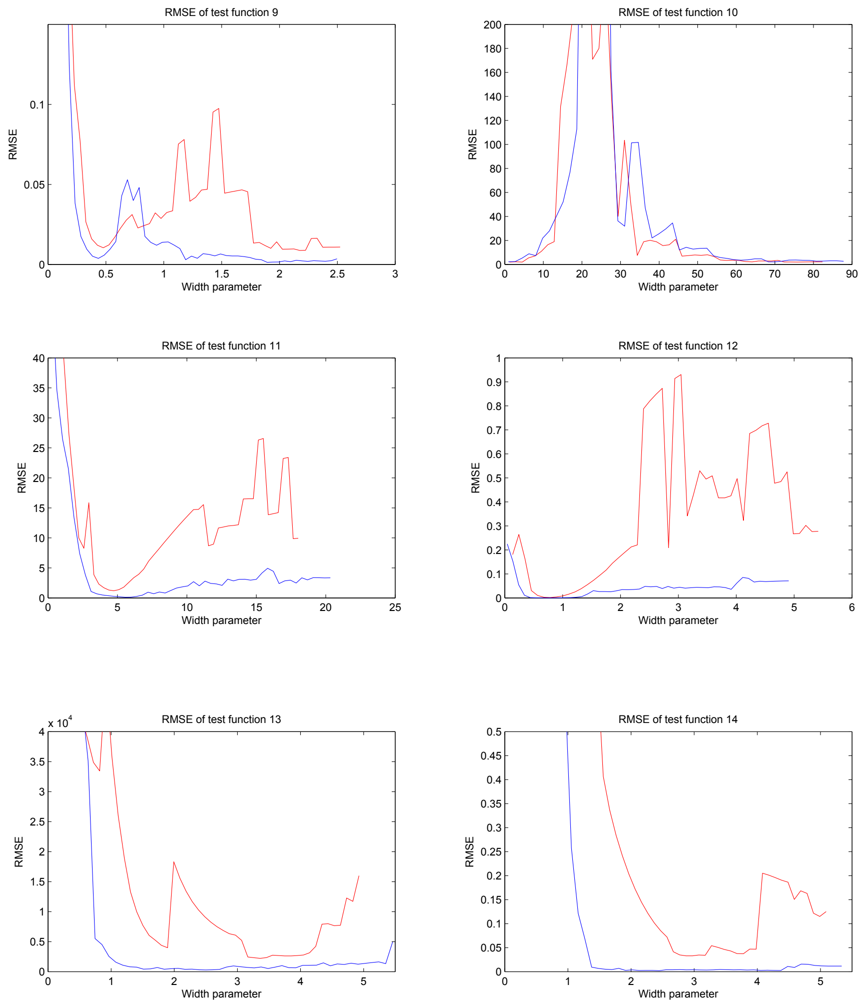


Figure A.6: Influence of the width parameter on the accuracy of the metamodel (continued)

somewhere just before the middle of the minimum and maximum value of the width parameter. This is not true for the metamodels fit on the small dataset of function 9,11 and 12, which are considered to be exceptions.

The default value for the width parameter of MATLAB is 1. Obviously, this can be improved by involving the width of the input space. Therefore, the mean of the minimum and maximum value, as mentioned before, is chosen to be the default value of `param.spread`. Note that varying the width parameter can yield better performances.

A.3 Support Vector Regression

As mentioned in in Section 3.4, SVR metamodels can be build with different kernels. These kernels have all a different influence at the performance of the metamodel. Some of these kernels take parameters to determine the form of the kernel. In this section, each kernel is tested to decide a default value of these parameters. The following kernels take parameters:

- Polynomial
- Gaussian
- Exponential
- Sigmoid (takes two parameters)
- Fourier
- B-Splines

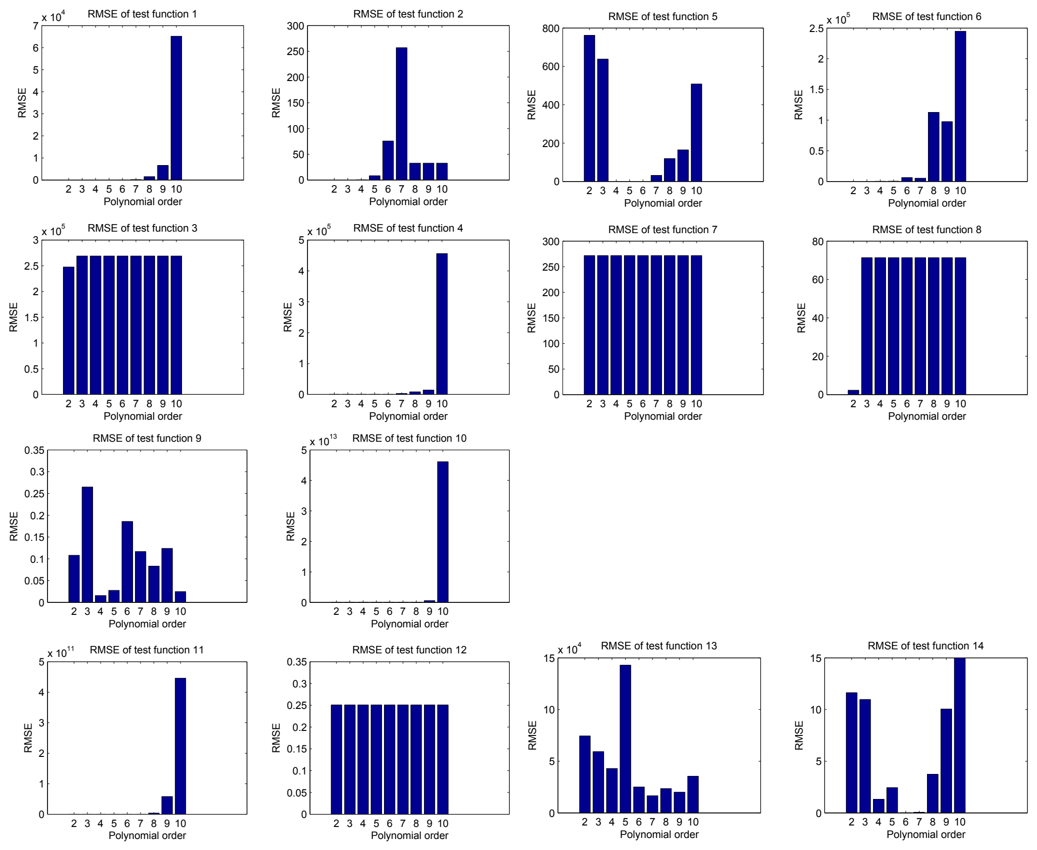
In the following the optimal value for the parameters of these kernels are tested by fitting metamodels of two datasets on 14 test functions, and calculating the RMSE for each metamodel with an additional validation set of 1000 points.

A.3.1 Polynomial

As mentioned in Section 3.4, the polynomial kernel of the SVR metamodels has the following form

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x} \cdot \boldsymbol{x}' \rangle^{d}. \tag{A.2}$$

The parameter of the polynomial kernel is thus the order d of the dot-product. This parameter can be set by `param.svr_poly_p1`. The parameter is varied from order 2 to 10. The plots of the tests are shown below.



All orders have the same performance with test functions 3, 7, 8 and 12. Only a second order polynomial kernel seem to make a little difference in these test functions. Test function 1, 4, 6 and 11 have all an ascending RMSE along with an ascending polynomial order. Test function 5 and 9 seem to reach their minimal RMSE at a polynomial order of 4, whereas test functions 13 14 have their minimum at an order of 7. Overall, a polynomial order of 4 seem to have a good overall performance, due to their minimum in test functions 5 and 9 and a good result in test functions 13 and 14. So the default value for `param.svr_poly_p1` is set to 4.

A.3.2 Gaussian and Exponential

The Gaussian kernel of the SVR metamodels has the following form

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\sigma^2}\right), \tag{A.3}$$

and the exponential kernel of the SVR metamodels has the following form

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||}{2\sigma^2}\right). \tag{A.4}$$

This parameter σ can be set by `param.svr_rbf_p1` for the Gaussian kernel and by `param.svr_erbf_p1`. These two kernels are both radial basis functions. Therefore notice is taken from the conclusions made in the section before. These kernels of the SVR metamodels are a transformation of the width parameter of the radial basis functions mentioned in the sector before. Therefore, the same default value is taken here as concluded in that section. The default value for `param.svr_rbf_p1` and `param.svr_erbf_p1` is thus a transformation of the mean of the maximum and minimum width of the input space.

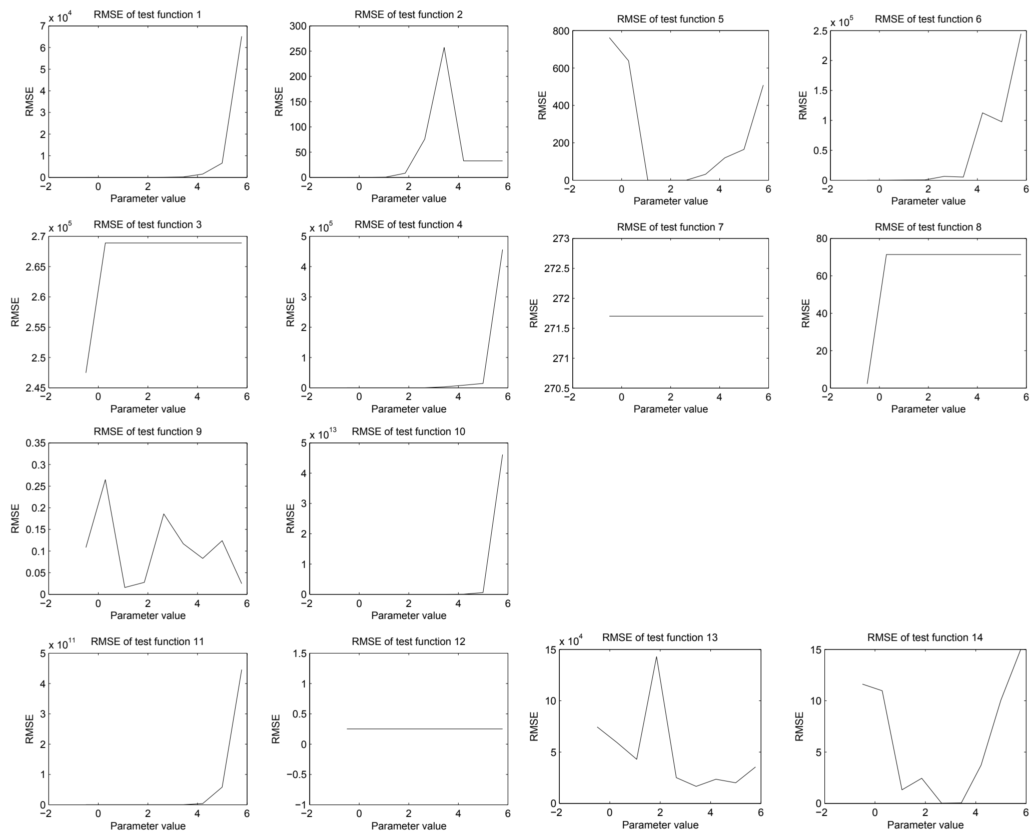


Figure A.7: Performance of different value for parameters of the fourier kernel of the SVR metamodel

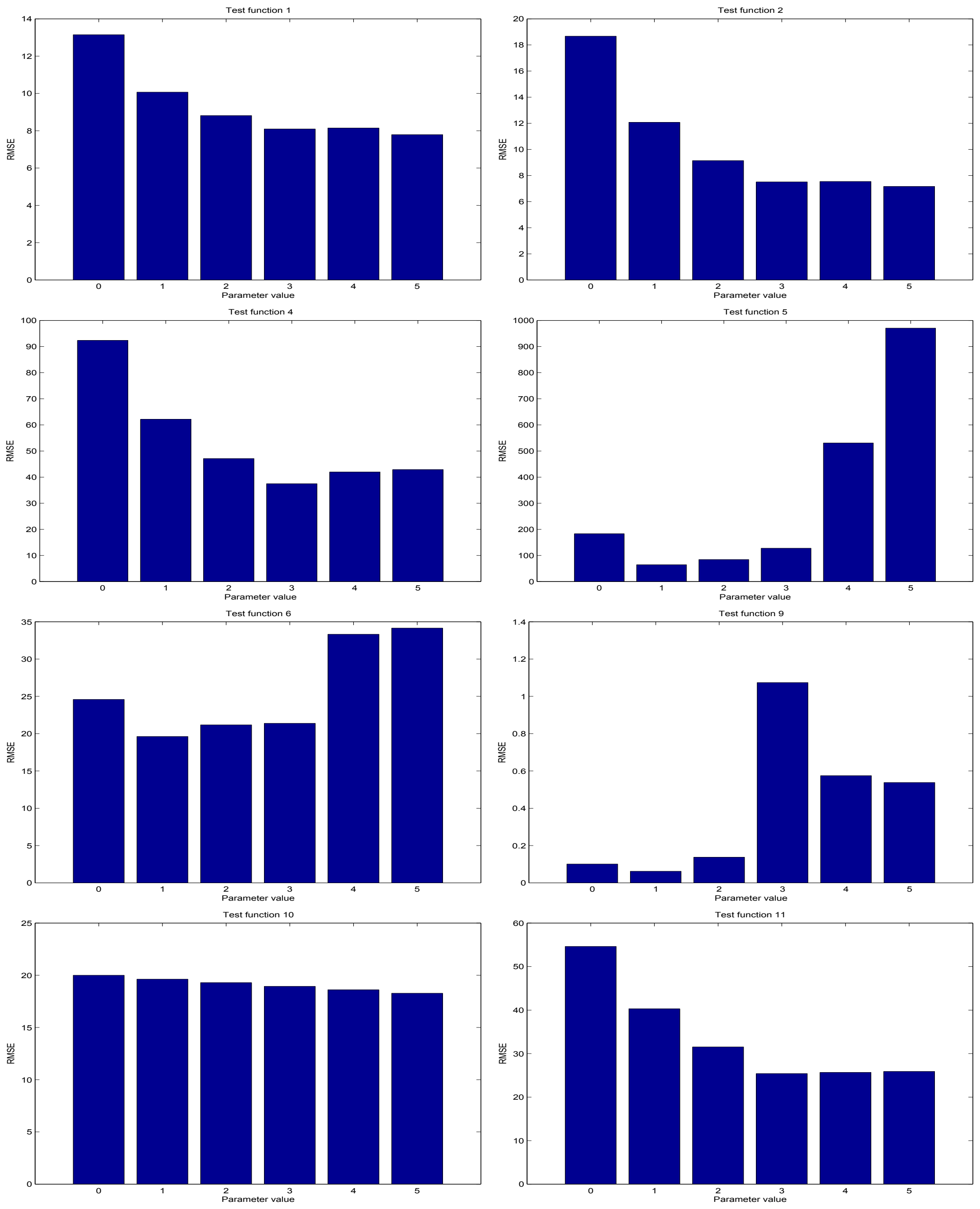


Figure A.8: Performance of different values for parameters of the B-Splines kernel of the SVR metamodel

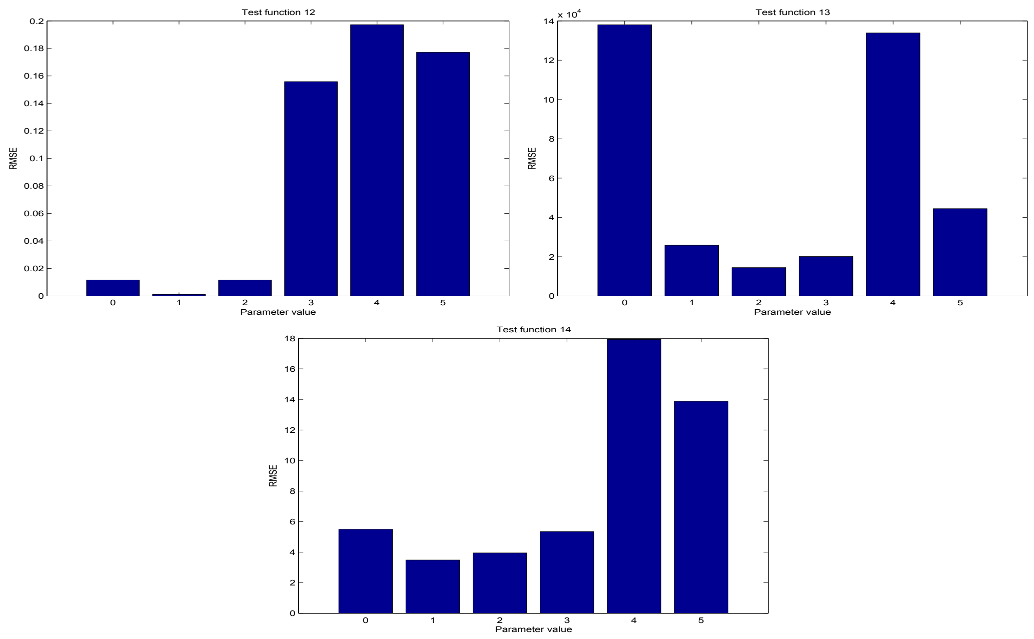


Figure A.9: Performance of different values for parameters of the B-Splines kernel of the SVR metamodel (continued)

A.3.3 Sigmoid

The Sigmoid kernel of the SVR metamodels has the following form

$$k(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\rho\langle \boldsymbol{x} \cdot \boldsymbol{x}' \rangle + \theta). \tag{A.5}$$

It is a special case, because this is the only kernel which takes two parameters. Due to the form of the function, the parameters will be chosen so that the function is centred around the mean of the dot products.

A.3.4 Fourier

The Fourier kernel of the SVR metamodels has the following form

$$k(\boldsymbol{x}, \boldsymbol{x}') = \frac{\sin(N + \frac{1}{2})(\boldsymbol{x} - \boldsymbol{x}')}{\sin(\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}'))} \tag{A.6}$$

The parameter of the Fourier kernel can be considered as an expansion in the $2N+1$ dimensional feature space. The parameter N is considered on the interval $[-0.5, 2\pi - 0.5]$ and can be set with `param.svr_fourier_p1`. The parameter is varied in $[-0.5, 2\pi - 0.5]$. The plots of the tests are shown in figure A.7.

Note that the RMSE's are very high. This is because its regularisation capability is poor, which is evident by consideration of its Fourier transform [45]. Chosen is for a default value of -0.5 for `param.svr_fourier_p1`, because this value has an overall low value.

A.3.5 B-Splines

The B-Splines kernel of the SVR metamodels has the following form

$$k(\boldsymbol{x}, \boldsymbol{x}') = B_{2N+1}(\boldsymbol{x} - \boldsymbol{x}') \tag{A.7}$$

The parameter of the B-Splines is an integer, which denotes $2N + 1$, and can be set with `param.svr_bsplines_p1`. The parameter is varied from 0 to 5. The plots of the tests are shown in figure A.8 and A.9 . Note that test functions 3,7 and 8 are not shown because these tests yield indecisive results.

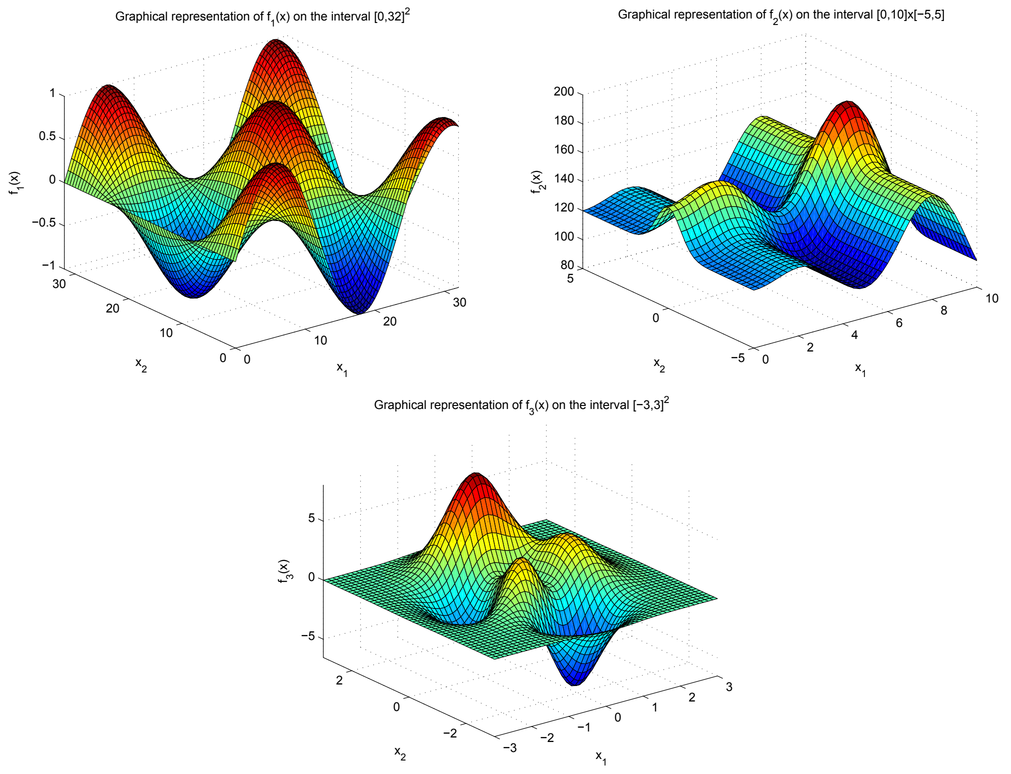


Figure A.10: Plots of the test functions considered for comparing kernel function

A.3.6 Comparison kernel functions

The kernel functions are tested on three different mathematical problems. These are all two-dimensional mathematical problems, the first two are taken from a paper by Jin, Chen and Simpson and the last one is the MATLAB-peaks function. They are given by

$$f_1(x) = \sin(\pi x_1/12)\cos(\pi x_2/16) \tag{A.8}$$

$$f_2(x) = (30 + x_1 \sin x_1)(4 + e^{-x_2^2}) \tag{A.9}$$

$$f_3(x) = 3(1 - x_1)^2 e^{-x_1^2-(x_2+1)^2} - 10(x_1/5 - x_1^3 - x_2^5)e^{-x_1^2-x_2^2} \tag{A.10}$$

$$- 1/3e^{-(x_1+1)^2-x_2^2} \tag{A.11}$$

The test functions are considered on the intervals shown in the plots in figure A.10.

The Support Vector Regression metamodel is trained with a dataset of 50 points, which are sampled using a Latin hypercube design. 1000 test points are used then for validation, which is also sampled using Latin hypercube design. Then the RMSE is calculated for each kernel function on these validation points. This is done for all three test problems.

The following figures shows the accuracy of each kernel functions used in SVR. The accuracy is measured in RMSE.

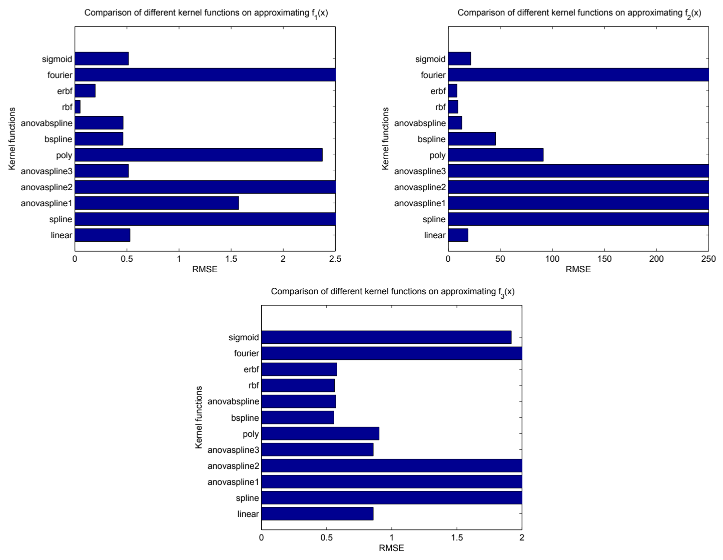


Figure A.11: Performance of the different kernels of the SVR metamodel on the tree test functions

As can be seen the Gaussian RBF and the Exponential RBF kernel functions seems to perform best in all test problems. Only in the last test problem the both b-splines kernel functions perform better. These kernel functions also perform good in all problems. The fourier, anovaspline1, anovaspline2 and spline seem not to be suitable for regression.

A special case is the sigmoid kernel function. This kernel function takes two parameters, when these two parameters are (close to) zero, the sigmoid kernel function approximates the linear kernel function. However, these parameters of zero or close to zero seems to be optimal. Therefore, the tuning of these parameters needs to be further investigated.

The linear, poly and anovaspline3 kernel function perform mediocre. They don't outperform the other functions but don't perform that bad.

A.4 MARS

Multivariate adaptive regression splines (MARS) metamodels can also be fitted with `MF_Fit`. This metamodel takes several parameters, namely

1. `model.maxdepth`,
2. `model.max_interactions`, and
3. `model.selfinteraction`.

The MARS metamodel is determined by its basis functions. The maximum number of basis functions $M_{\max}$ is set by `model.maxdepth`. More basis functions will provide a more extensive and (probably) accurate metamodel. However, this will cause a higher computational time, because more basis functions need to be calculated. The question if a large model is desirable over a small model can also arise. Therefore, a good trade-off must be made between complexity, for good approximation, and usability, by reducing computational time.

The basis set of 14 test functions is used to determine an optimal default value for
`model.maxdepth`. Each test function is fitted with multiple MARS metamodels, with an ascending `model.maxdepth` from 1 to 20. Two randomly generated datasets are used here, a big dataset consisting of 100 samples and a small dataset consisting of 50 samples.

As mentioned earlier the computational time will increase together with the complexity, i.e. `model.maxdepth`. To illustrate this, the average computational time needed to build the metamodels, the average over all the test functions, is plotted against the maximum number of basis functions $M_{\max}$ of the metamodels, `model.maxdepth` in figure A.12.

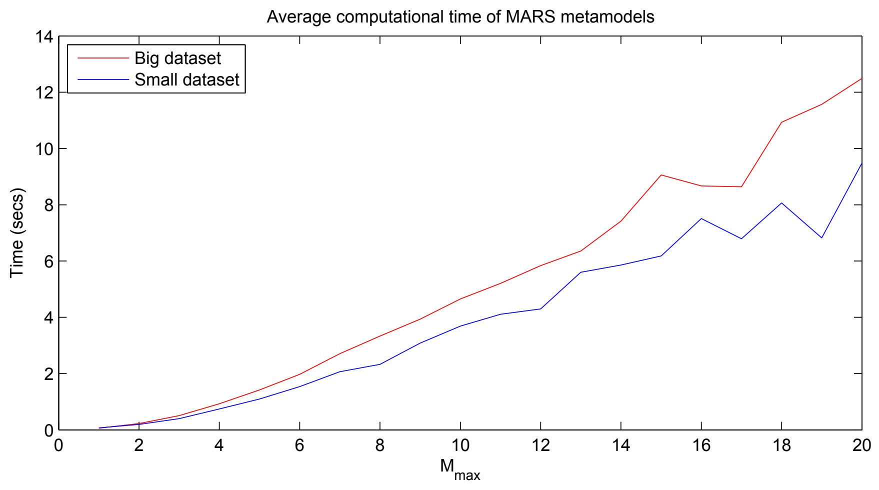


Figure A.12: Time needed for building the MARS metamodel with an increasing depth

As can be seen, for both datasets a more or less linear relation exists between the maximum number of basis functions and the needed computational time to build the model. As the maximum number of basis functions increase the needed computational time to build to model also increases. This is because more basis functions will be calculated and evaluated, what leads to more computational time. Obviously, a smaller dataset takes less computational time as less calculations has to be done. As comparison, polynomial and kriging models and radial basis functions take less than a second.

A metamodel with high complexity does not necessarily mean an accurate approximation. Therefore, the metamodels are evaluated with an additional dataset of 1000 samples. The predictions of the metamodels are compared with the actual values and the root mean square error is calculated. The metamodels of test functions 3 and 7 are omitted, because the metamodels have the same predictions for each `model.maxdepth`, i.e. each metamodel has a depth of 1. The results are shown in the plots given in figure A.13 and A.14.

As expected, the RMSE decreases as the $M_{\max}$ increases. The RMSE's of test functions 1,2 and 4 seem to stabilize after $M_{\max} = 6$. Note that these test functions are linear. All other test functions stabilize around $M_{\max} = 10$, after $M_{\max} = 10$ the RMSE oscillates around the same level. Test functions 6 and 7 are not considered, because test function 6 oscillates too much to make any conclusion and the models of test function 7 are equal, hence the equal RMSE.

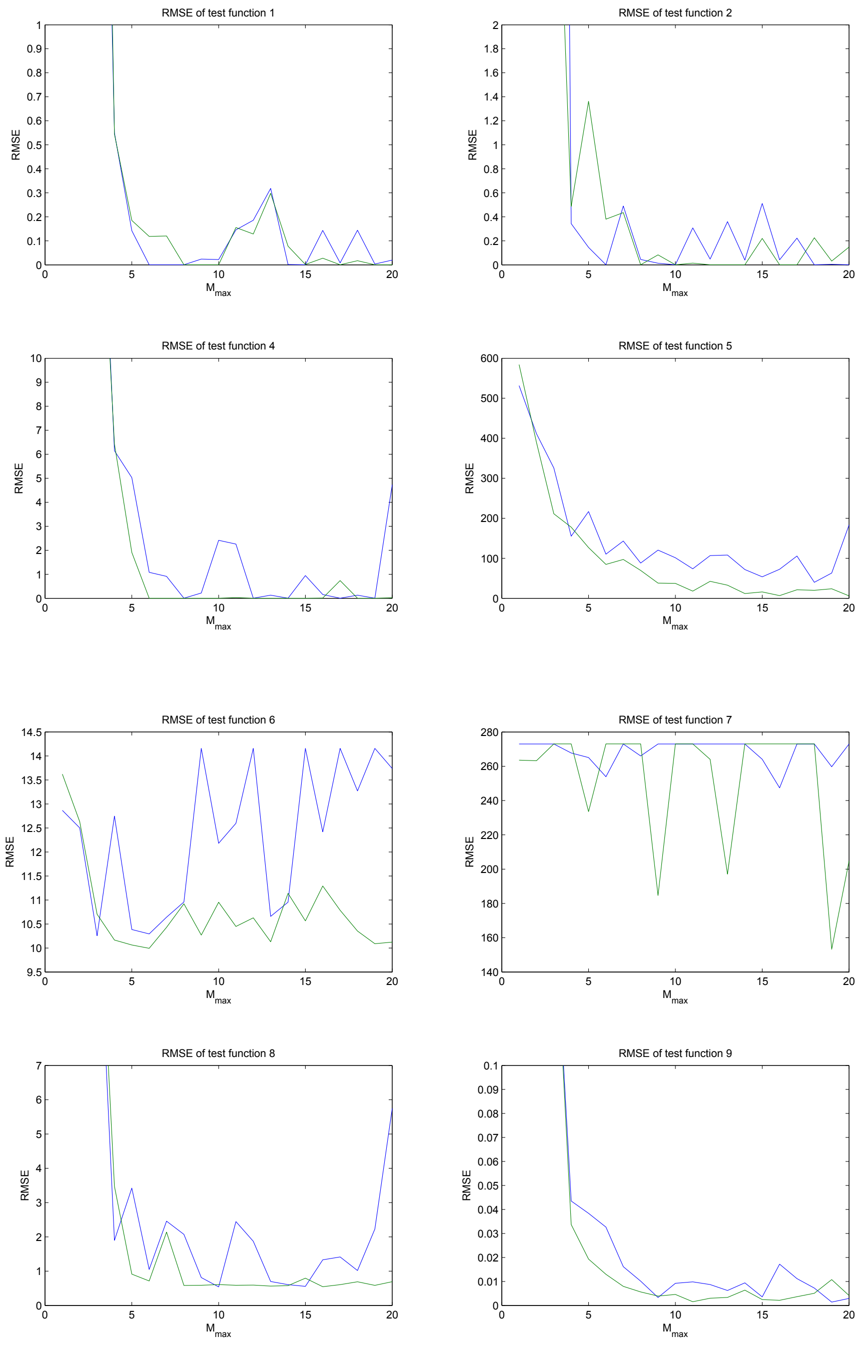


Figure A.13: Performance of the MARS metamodels on different test functions

The conclusion can be made that a value of 10 for `model.maxdepth` is desirable. This value correspond to an average computational time of circa 4.7 seconds for a dataset of 100 samples and circa 3.7 seconds for 50 samples. This is significant more than other methods like polynomial models or kriging, but is still acceptable. The default value of `model.maxdepth` is therefore set to 10.

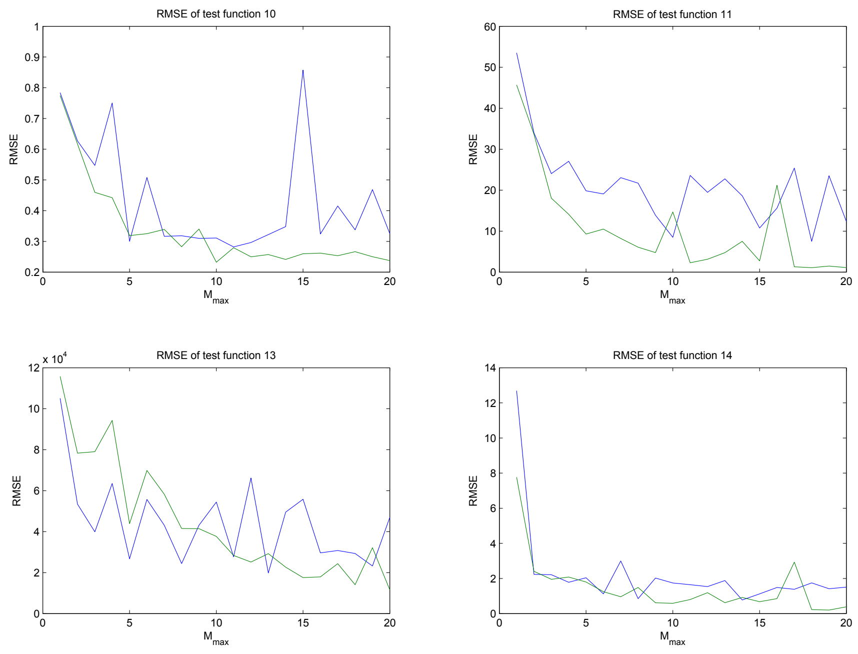


Figure A.14: Performance of the MARS metamodels on different test functions (continued)

Another parameter which influences the performance of the MARS metamodel is
`model.max_interactions`. As explained in Sector 3.5, as well as univariate as interaction basis functions exists. Interaction basis function involves basis function of multiple explanatory variables. When no interactions are desired `model.max_interactions` is set to 0. Only univariate basis functions are then considered. A value of 1 signifies an interaction basis function of 2 explanatory variables (1 interaction between 2 variables). To consider a full model, the parameter `model.max_interactions` is set to its maximum value, i.e. the number of explanatory variables minus 1. If the number of explanatory variables is large, this will cause an increase in the computational

time, because more basis functions needs to be evaluated. The default value will be set to its maximum value, because a full model, i.e. with all possible interactions, is desirable.

The last parameter involved in building the MARS metamodel is the parameter
`model.selfinteraction`. This parameter can take logical values, which means the value can be set to `true` or `false`. When this parameter is set to `false`, basis functions cannot be split into dimensions that are already used in that or previous knot(s), meaning that consecutive knots cannot split the same dimension. No self-interactions causes a smaller model with less computational time. A full MARS metamodel is desirable, so the default value is `true`.

Appendix B

Metamodelling confidence intervals

With the use of metamodels predictions can be made of unknown datapoints. These prediction provide an indication for the actual value, because these metamodels are merely approximations. So, the predictions given by the metamodels contain a degree of uncertainty, which results in a residual, an error of approximation. Unfortunately, most metamodels provide only a prediction without an indication of uncertainty. An option can be to provide additional information in form of a confidence interval.

Classical notions of confidence intervals, for example based on least-squares residuals, are not applicable [39]. So alternative methods must be searched or invented.

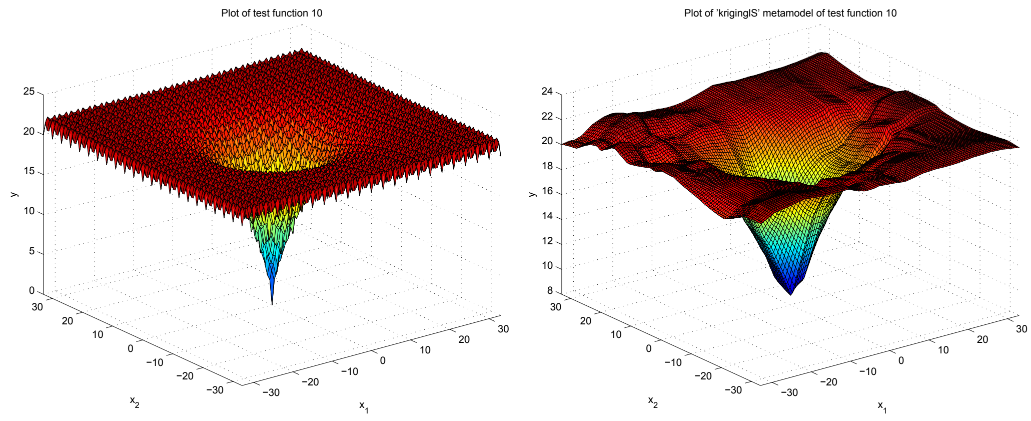


Figure B.1: Original function and the metamodel, both plotted on the domain

Together with the predictions of a leave-one-out cross-validation, an error of approximation is obtained. In leave-one-out cross-validation an error of approximation is always obtained, because an unknown point that is left out of the original dataset, is predicted. A possible alternative method is providing a confidence interval based on the errors of approximation.

The error of approximation of an unknown point should be predicted. This can be done by fitting a metamodel on these errors of approximation. This residual metamodel is then able to predict an error of approximation of an unknown point.

In order to prevent interference of negative and positive errors of approximation, the absolute error of approximation is taken as input for the residual metamodel. The negative and positive errors of approximation could also be split, so two sets of errors of approximation obtained. This way a negative and a positive residual prediction can be given, causing a confidence interval. This is not preferable, because two metamodels must be fitted, both based on a part of the dataset. In contrast, when considering absolute errors of approximation, one metamodel can be build, based on a full dataset. This way, the dataset is larger and thus the metamodel is more dependable.

Considering a residual metamodel predicting the error of approximation for an unknown point based on absolute values, the prediction of the error of approximation acts both as a positive and as a negative range.

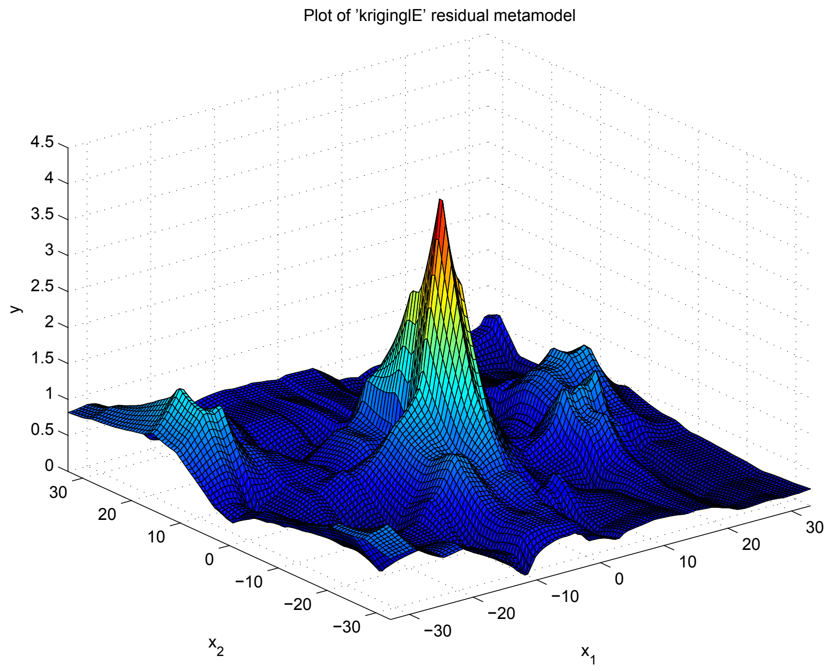


Figure B.2: The residual metamodel plotted on the domain

This idea will be illustrated by applying the idea to test function 10. A dataset is generated of 100 datapoints for training and a dataset of 1000 datasets is generated for validating. First off, a metamodel is build, the metamodel with the lowest MAPE, based on leave-one-out cross-validation, is chosen. This metamodel is the kriging metamodel with a linear regression function and a spherical correlation function. The metamodel and the original function are both shown in figure B.1.

The leave-one-out cross-validation yields predictions of the training dataset that are not accurate, i.e. has an error of approximation. The absolute difference between the real values and the prediction are the absolute residuals. A metamodel is again fitted, but this time on the absolute residuals in combination with the datapoints. Again, the metamodel with the lowest RMSE, based on leave-one-out cross-validation, is chosen. This metamodel is the kriging metamodel with a linear regression function and a exponential correlation function.

So, a 'kringinglS' metamodel is fitted for predicting the response values is fitted, based on the original dataset, and an additional residual 'kringinglE' metamodel for predicting the error of approximation is fitted, based on the error of approximation of the leave-one-out cross-validation predictions of the 'kringinglS' metamodel.

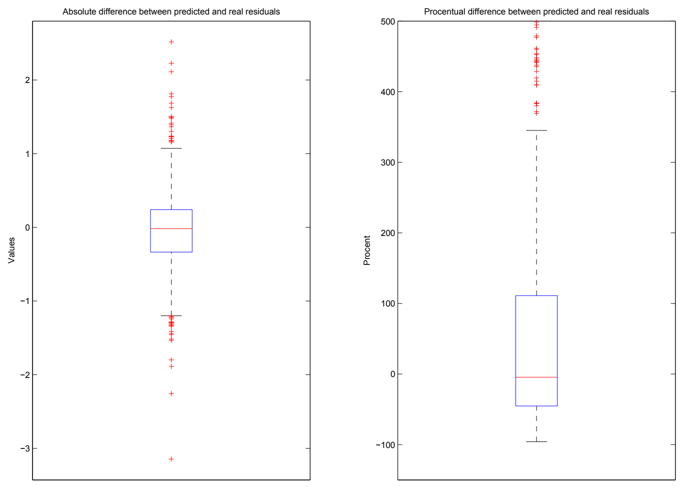


Figure B.3: Boxplots of absolute and procentual differences between the predicted and real residuals

As can be seen, the residual metamodel reaches a top around $(0, 0)$, meaning the residual metamodel predicts a large residual here. The original metamodel is indeed not accurate in this region as can be seen in Figure B.1, so the residual metamodel gives a good indication.

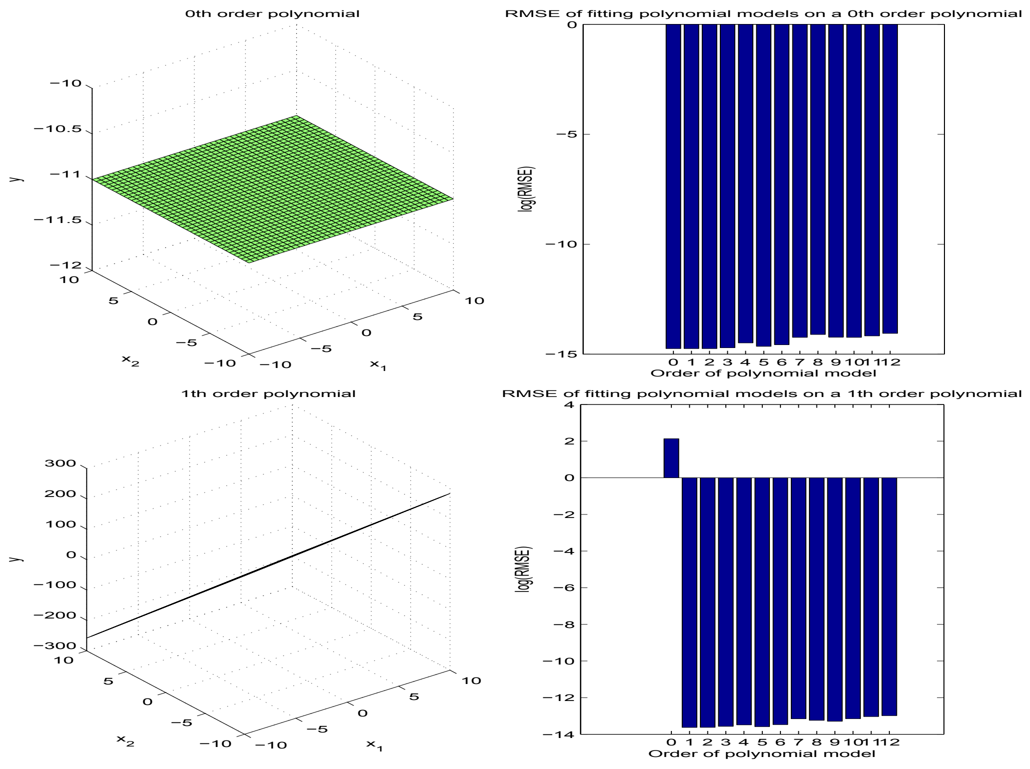
However, to validate this residual metamodel, the validation dataset containing 1000 datapoints is predicted by the original metamodel and the residual metamodels. This yields two predictions, one for the observation value and one for the error of approximation. The prediction for the observation value is compared with the real observation value, yielding an error of approximation.

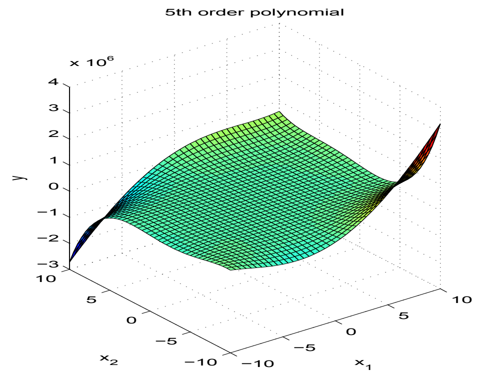
The absolute difference between the prediction of the residual and the actual residual is around zero. However, in account must be taken that the residuals of this test function are small values and are bound to be around zero. Therefore, a relative procentual difference is calculated, which is based on the real residual. The difference between the prediction and the actual value of the residual is divided by the actual value of the residual, which yields a procentual value.

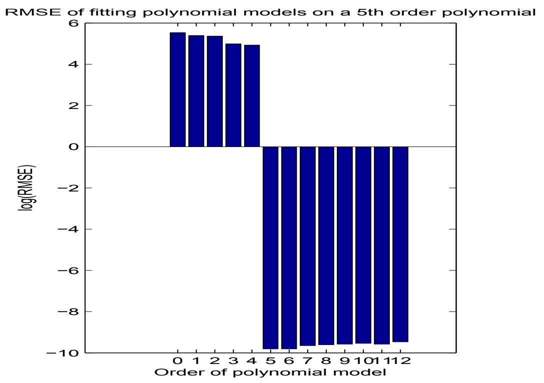
Obviously, the predictions of residual are not always accurate. Also, not much can be said over the reliability of the predictions of the residuals. However, they provide a good indication and can provide additional information.

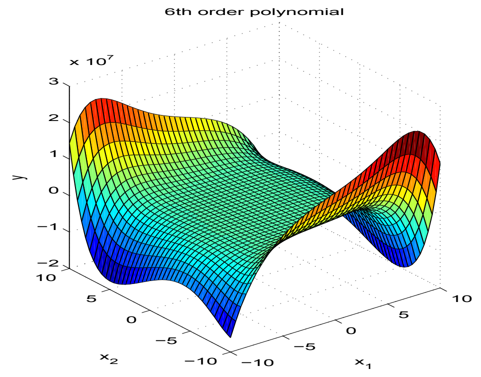
Appendix C

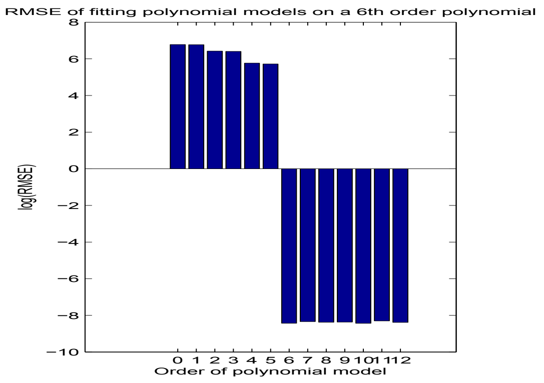
Additional figures of polynomial metamodels of polynomial functions

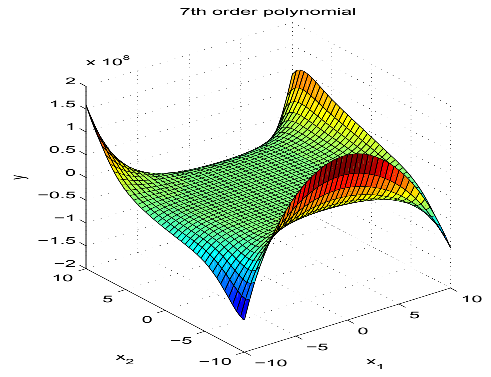


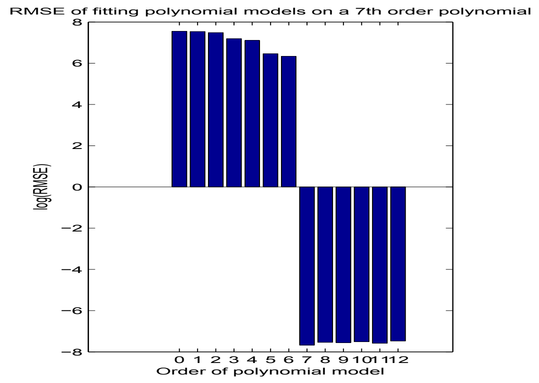

5th order polynomial
x 10^6
y
x_1
x_2

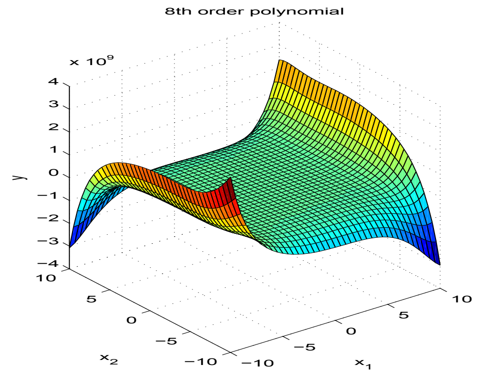

RMSE of fitting polynomial models on a 5th order polynomial
log(RMSE)
Order of polynomial model
0 1 2 3 4 5 6 7 8 9 10 11 12

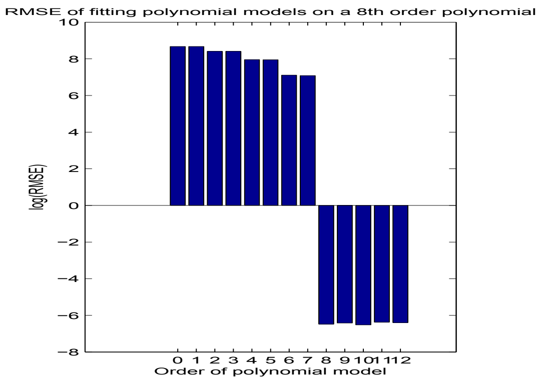

6th order polynomial
x 10^7
y
x_1
x_2

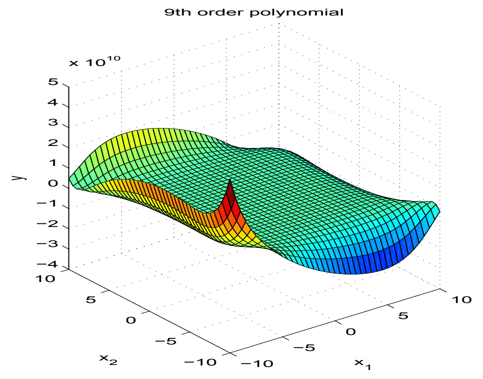

RMSE of fitting polynomial models on a 6th order polynomial
log(RMSE)
Order of polynomial model
0 1 2 3 4 5 6 7 8 9 10 11 12

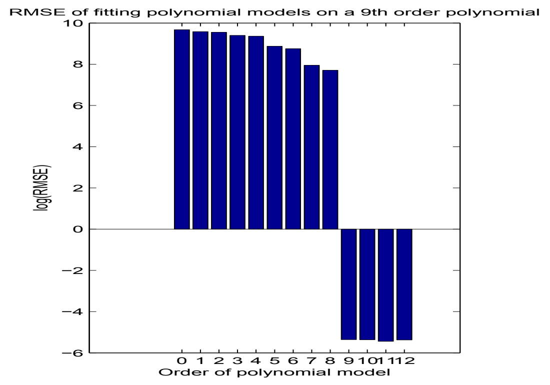

7th order polynomial
x 10^8
y
x_1
x_2

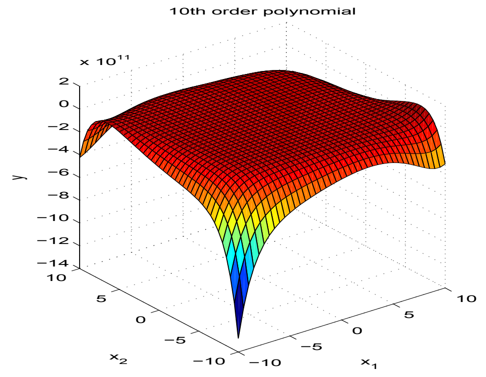

RMSE of fitting polynomial models on a 7th order polynomial
log(RMSE)
Order of polynomial model
0 1 2 3 4 5 6 7 8 9 10 11 12

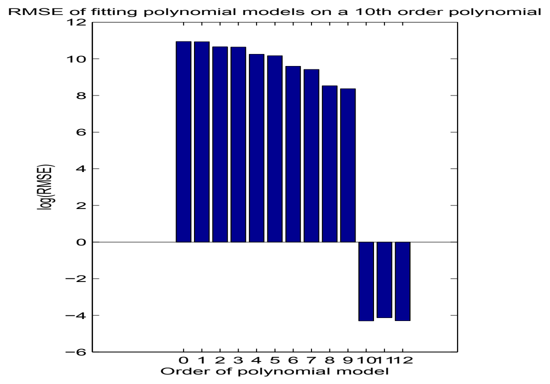

8th order polynomial
x 10^9
y
x_1
x_2

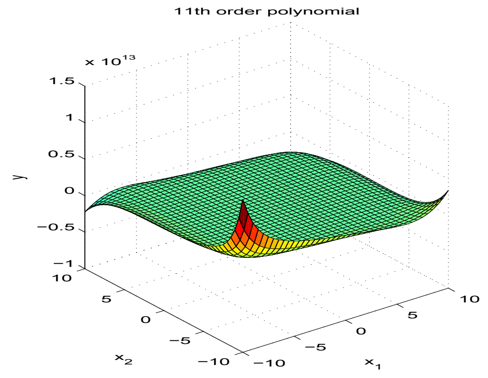

RMSE of fitting polynomial models on a 8th order polynomial
log(RMSE)
Order of polynomial model
0 1 2 3 4 5 6 7 8 9 10 11 12

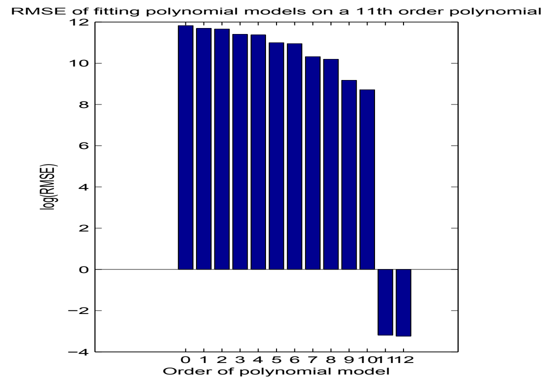

9th order polynomial
x 10^10
y
x_2
x_1

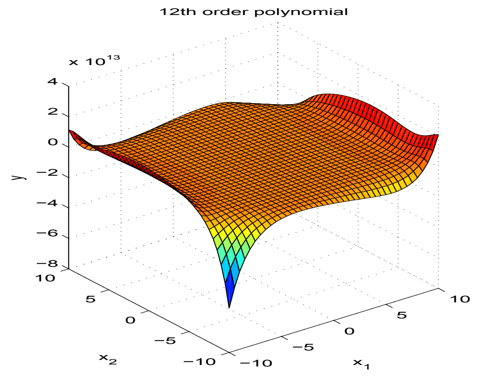

RMSE of fitting polynomial models on a 9th order polynomial
log(RMSE)
Order of polynomial model

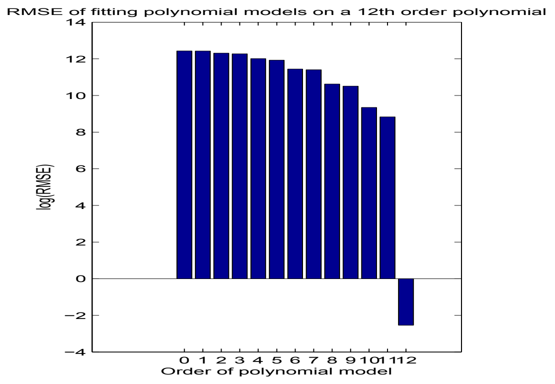

10th order polynomial
x 10^11
y
x_2
x_1


RMSE of fitting polynomial models on a 10th order polynomial
log(RMSE)
Order of polynomial model


11th order polynomial
x 10^13
y
x_2
x_1


RMSE of fitting polynomial models on a 11th order polynomial
log(RMSE)
Order of polynomial model


12th order polynomial
x 10^13
y
x_2
x_1


RMSE of fitting polynomial models on a 12th order polynomial
log(RMSE)
Order of polynomial model