Linköping Studies in Science and Technology
Thesis No. 1565

Metamodel-Based Design Optimization

– A Multidisciplinary Approach for Automotive Structures

Ann-Britt Ryberg

Linköping University
INSTITUTE OF TECHNOLOGY

LIU-TEK-LIC-2013:1

Division of Solid Mechanics
Department of Management and Engineering
Linköping University
SE-581 83 Linköping, Sweden

January 2013

Cover:
Illustration of metamodel-based multidisciplinary design optimization with four different loadcases. More information about this specific example is found in Section 6.3.

Printed by:
LiU-Tryck, Linköping, Sweden, 2013
ISBN 978-91-7519-721-0
ISSN 0280-7971

Distributed by:
Linköping University
Department of Management and Engineering
SE-581 83 Linköping, Sweden

Preface

The work presented in this thesis has been carried out at Saab Automobile AB and Combitech AB in collaboration with the Division of Solid Mechanics, Linköping University. It has been partly sponsored by the Swedish governmental agency for innovation systems (VINNOVA/FFI) in the project “Robust and multidisciplinary optimization of automotive structures”, and it has also been a part of the SFI/ProViking project ProOpt.

I would like to thank my supervisor Professor Larsgunnar Nilsson for his encouraging guidance throughout the course of this work. A very special appreciation also goes to my PhD student colleague Rebecka Domeij Bäckryd for our close collaboration and very fruitful discussions.

Additionally, special thanks to my manager Tomas Sjödin for being one of the initiators of the project and always supporting me. Likewise, I am very thankful to Gunnar Olsson for helping me to continue my research after the bankruptcy of Saab Automobile AB.

I am also grateful to all my colleagues, friends and family for their support and interest in my work. Finally, I would like to especially thank my beloved fiancé Henrik and dedicate this work to our coming miracle!

Abstract

Automotive companies are exposed to tough competition and therefore strive to design better products in a cheaper and faster manner. This challenge requires continuous improvements of methods and tools, and simulation models are therefore used to evaluate every possible aspect of the product. Optimization has become increasingly popular, but its full potential is not yet utilized. The increased demand for accurate simulation results has led to detailed simulation models that often are computationally expensive to evaluate. Metamodel-based design optimization (MBDO) is an attractive approach to relieve the computational burden during optimization studies. Metamodels are approximations of the detailed simulation models that take little time to evaluate and they are therefore especially attractive when many evaluations are needed, as e.g. in multidisciplinary design optimization (MDO).

In this thesis, state-of-the-art methods for metamodel-based design optimization are covered and different multidisciplinary design optimization methods are presented. An efficient MDO process for large-scale automotive structural applications is developed where aspects related to its implementation is considered. The process is described and demonstrated in a simple application example. It is found that the process is efficient, flexible, and suitable for common structural MDO applications within the automotive industry. Furthermore, it fits easily into an existing organization and product development process and improved designs can be obtained even when using metamodels with limited accuracy. It is therefore concluded that by incorporating the described metamodel-based MDO process into the product development, there is a potential for designing better products in a shorter time.

Keywords: metamodel-based design optimization (MBDO); multidisciplinary design optimization (MDO); automotive structures

List of Papers

In this thesis, the following papers have been appended:

I. R. D. Bäckryd, A.-B. Ryberg, and L. Nilsson (2013). *Multidisciplinary design optimization methods for automotive structures*, Submitted.

II. A.-B. Ryberg, R. D. Bäckryd, and L. Nilsson (2013). *A metamodel-based multidisciplinary design optimization process for automotive structures*, Submitted.

Own contribution

The work resulting in the two appended papers have been a joint effort by Rebecka Domeij Bäckryd and me. My contribution to the first paper includes being an active partner during the writing process. As for the second paper, I have had the main responsibility for writing the paper and conducting the application example.

Contents

Preface iii

Abstract v

List of Papers vii

Contents ix

Part I – Theory and Background

1 Introduction 3

2 Optimization 5

2.1 Structural Optimization 6

2.2 Metamodel-Based Design Optimization. 6

2.3 Multi-Objective Optimization 6

2.4 Probabilistic-Based Design Optimization 7

2.5 Multidisciplinary Design Optimization 7

3 Automotive Product Development 9

3.1 Multidisciplinary Design Optimization of Structures 10

4 Metamodel-Based Design Optimization 11

4.1 Design of Experiments. 12

4.1.1 Latin Hypercube Sampling 13

4.1.2 Distance-Based Designs 15

4.1.3 Low-Discrepancy Sequences 15

4.1.4 Sampling Size and Sequential Sampling 16

4.2 Screening 16

4.3 Metamodels 17

4.3.1 Kriging 20

4.3.2 Radial Basis Functions 22

4.3.3 Artificial Neural Networks . . . 24
4.3.4 Multivariate Adaptive Regression Splines . . . 30
4.3.5 Support Vector Regression . . . 32
4.4 Metamodel Validation . . . 35
4.4.1 Error Measures . . . 35
4.4.2 Cross Validation . . . 37
4.4.3 Generalized Cross Validation and Akaike's Final Prediction Error . . . 39
4.5 Optimization Algorithms . . . 40
4.5.1 Evolutionary Algorithms . . . 41
4.5.2 Particle Swarm Optimization . . . 45
4.5.3 Simulated Annealing . . . 46

5 Multidisciplinary Design Optimization 49
5.1 Single-Level Methods . . . 50
5.2 Multi-Level Methods . . . 50
5.3 Suitable Methods for Automotive Structures . . . 53

6 An MDO Process for Automotive Structures 55
6.1 Requirements . . . 55
6.2 Process description . . . 56
6.3 Application Example . . . 59

7 Discussion 65

8 Conclusion and Outlook 67

9 Review of Appended Papers 69

Bibliography 71

Part II – Appended Papers

Paper I
Multidisciplinary design optimization methods for automotive structures. . . . 81
Paper II
A metamodel-based multidisciplinary design optimization process for automotive structures . . . 107

Part I

Theory and Background

1 Introduction

Automotive companies work in a strongly competitive environment and continuously strive to design better products in a cheaper and faster manner. This is a challenge that requires continuous improvements of methods and processes. Automotive development has thus gone from a trial and error approach in a hardware environment to completely rely on computer aided engineering (CAE). The number of prototypes is kept to a minimum in order to reduce cost and development time. Instead, every possible aspect of the product is evaluated using detailed simulation models. These detailed models are often computationally expensive to evaluate, which is a challenge when many evaluations are needed, as when performing optimization or robustness studies. One way to ease the computational burden can be to use approximations of the detailed simulation models that take little time to evaluate. One approach is to build metamodels, i.e. surrogate models developed based on a series of simulations using the detailed models. The idea originates from fitting a surrogate model to a series of designed physical experiments, see e.g. Myers et al. (2008). The methods related to metamodels and their applications have been extensively investigated and developed over the years. The use of metamodels during optimization studies, so-called metamodel-based design optimization, has been proven to be efficient in many cases. However, the topic is still an active area of research.

Many groups with different responsibilities are involved during the development of a new product. These groups need to work concurrently and autonomously for the development to be efficient. However, the groups must also cooperate closely to ensure that the product meets all the requirements. The term “group” is used here to denote both the administrative unit and a team working with a specific task. Traditionally, the goal during automotive development has been to find a feasible design, i.e. a design that fulfils all defined requirements, not necessarily an optimum one. When optimization has been used, it has commonly been applied by one group at a time, requiring the design to be checked and adjusted to meet the requirements from other groups. A better strategy would be to use multidisciplinary design optimization (MDO), which is a methodology for optimizing several disciplines, or performance aspects, simultaneously and taking the interactions between the disciplines into account. Since the responsibility for the performance is distributed, MDO usually involves several groups. The potential of multidisciplinary design optimization have been recognized, but MDO has not yet been integrated within automotive product development due to several challenges. It needs to suit the company

organization and fit into the product development process, which places restrictions on the choice of method. Furthermore, MDO includes evaluations of several different detailed simulation models for a large number of variable settings, which requires considerable computer resources.

The VINNOVA/FFI project *"Robust and multidisciplinary optimization of automotive structures" (Swedish: "Robust optimering och Multidisciplinär optimering av fordons strukturer")* was established to find suitable methods for implementing robust and multidisciplinary design optimization in automotive development. The multidisciplinary aspect is the focus in this thesis, and the goal has been to develop an efficient MDO process for large-scale structural applications. The methodology takes the special characteristics of automotive structural applications into account as well as considers aspects related to implementation within an existing organisation and product development process.

The presented work is also a part of the SFI/ProViking project ProOpt which aims at developing methods for optimization-driven design. The objective to find an MDO process suitable for automotive structural applications fits perfectly also within the scope of that project.

The chapters following this introduction will introduce important optimization concepts and give a short description of automotive product development. Since the use of metamodels is an essential part of automotive structural optimization, the main part of this thesis is devoted to metamodel-based design optimization. After a general description of multidisciplinary design optimization methods, a presentation of an MDO process suitable for large-scale automotive structural applications is presented and demonstrated in a simple example. The thesis is then ended with a discussion regarding the presented MDO process, conclusions and an outlook on further needs.

2 Optimization

Optimization is a procedure for achieving the best possible solution to a specific problem while satisfying certain restrictions. A general optimization problem can be formulated as

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0} \\ & \mathbf{x}^{lower} \leq \mathbf{x} \leq \mathbf{x}^{upper} \end{aligned} \tag{2.1}$$

The goal is to find the design variables $\mathbf{x}$ that minimize the objective function $f(\mathbf{x})$. In general, the problem is constrained, i.e. there are a number of inequality and equality constraints represented by the vectors $g(\mathbf{x})$ and $h(\mathbf{x})$ that should be fulfilled. If the problem lacks constraints, the problem is said to be unconstrained. The design variables are allowed to vary between an upper and a lower limit, called $\mathbf{x}^{upper}$ and $\mathbf{x}^{lower}$ respectively, which defines the design space. The design variables can be continuous or discrete, meaning that they can take any value, or only certain discrete values, between the upper and lower limits. Design points that fulfil all the constraints are feasible, while all other design points are unfeasible.

The general formulation in Equation (2.1) can be reorganized into the simpler form

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \end{aligned} \tag{2.2}$$

In this formulation, the inequality constraints $\mathbf{g}(\mathbf{x})$ contain all three types of constraints in the former formulation. Each equality constraint is then replaced by two inequality constraints and included, together with the upper and lower limits of the design variables, in the constraint vector $\mathbf{g}(\mathbf{x})$. Both formulations can be used for maximization problems if the objective function $f(\mathbf{x})$ is multiplied with -1.

The solution to an optimization problem is called the optimum solution and this solution is usually found using some numerical technique. An iterative search process that uses information from previous iterations is then applied. When the objective and constraint functions are evaluated during the solution process, one or several analyzers are used. For a vector of design variables $\mathbf{x}$, an analyzer returns a number of responses denoted by $\mathbf{y}$.

These responses can be combined into the objective and constraint functions for that specific vector of design variables.

2.1 Structural Optimization

A structure is a collection of physical components that are arranged to carry loads. Optimization of structures is called structural optimization, and the analyzer is often a finite element (FE) model. For these cases, the state functions (governing equations) must be fulfilled, which can be seen as constraints to the optimization problem. Three types of structural optimization can be distinguished: size, shape, and topology optimization. In size optimization, the design variables represent structural properties, e.g. sheet thicknesses. In shape optimization, the design variables instead represent the shape of material boundaries. Topology optimization is the most general form of structural optimization and is used to find where material should be placed to be most effective.

2.2 Metamodel-Based Design Optimization

The detailed simulation models used for structural optimization are often computationally expensive to evaluate. Metamodel-based design optimization, in which metamodels are used for the evaluations, can then be an attractive approach to decrease the required computational effort. This is in contrast to direct optimization where the evaluations are done using the detailed simulation models directly.

Metamodels are approximations of the detailed simulation models that take little time to evaluate. They are developed based on a series of simulations using the detailed models, either iteratively during the course of the optimization process or before the solution of the optimization problem starts. Metamodels can be simple and valid only over a small portion of the design space. Others are more complex and intended to capture the response over the complete design space. Additionally, they can either interpolate or approximate the dataset used to develop the model. Interpolating metamodels are intuitively more appealing for deterministic simulations. However, interpolating metamodels are not necessarily better than approximating ones at predicting the response between the simulated points. Further, interpolating metamodels capture the numerical noise, while approximating ones can smooth it out. Metamodel-based design optimization is covered in more detail in Chapter 4.

2.3 Multi-Objective Optimization

The optimization problem defined in Equation (2.2) is a single-objective optimization

problem. It has one objective function that should be minimized. Many variants of this problem can be found. When solving multi-objective optimization (MOO) problems, two or more objective functions should be minimized simultaneously. The simplest approach is to convert the problem into a single-objective problem. This can be done by minimizing one of the objective functions, usually the most important one, and to treat all the others as constraints. Another way is to create a single objective function as a combination of the original objectives. Weight coefficients can then be used to mirror the relative importance of the original objective functions. The drawback of the aforementioned methods is that only one single optimum is found. If the designer wants to modify the relative importance of the objective functions in retrospect, the optimization process must be performed again. An alternative approach is to find a number of Pareto optimal solutions. A solution is said to be Pareto optimal if there exist no other feasible solution yielding a lower value of one objective without increasing the value of at least one other objective. The designer will then have a set of solutions to choose among, and the trade-off between the different objective functions can be performed after the optimization process has been carried out.

2.4 Probabilistic-Based Design Optimization

It can be important to deal with uncertainties in the design variables when a product is designed. In contrast to deterministic design optimization, these variations are considered when performing probabilistic-based design optimization. In robust design optimization, a product that performs well and is insensitive to variations in the design variables is sought. This can be achieved by making a trade-off between the mean value and the variation of the product performance. In reliability-based design optimization on the other hand, the probability distribution of the product performance is calculated. The probability of failure is typically constrained to be below a certain level. Large variation in the performance of the product can thus be allowed as long as the probability of failure is low.

2.5 Multidisciplinary Design Optimization

Multidisciplinary design optimization evolved as a new engineering discipline in the area of structural optimization, mainly within the aerospace industry (Agte et al., 2010). Multidisciplinary design optimization is used to optimize a product taking into account more than one discipline simultaneously. If the objective, constraints, or variables are related to different disciplines, the problem is consequently multidisciplinary. Giesing and Barthelemy (1998) provide the following definition of MDO: *"A methodology for the design of complex engineering systems and subsystems that coherently exploits the synergism of mutually interacting phenomena."* In general, a better design can be found when considering the interactions between different aspects of a product than when con-

sidering them as isolated entities. Different configurations, called loadcases, can be considered within each discipline. Each loadcase can be seen as a part of the MDO problem, i.e. a subspace. The MDO methodology can just as well be applied to different loadcases within one single discipline, and the problem is then not truly multidisciplinary. However, the idea of finding a better solution by taking advantage of the interactions between subspaces still remains.

Some of the variables in an MDO problem are related to several subspaces, while others are unique to one specific subspace. These variables are called shared and local variables, respectively. In the general case, the subspaces are coupled, i.e. output from one subspace is needed as input to another subspace. The couplings between subspaces are handled by the so-called coupling variables and iterative approach is needed to find a consistent solution, i.e. a solution in balance. This is referred to as multidisciplinary feasibility by Cramer et al. (1994), but since feasibility in an optimization context refers to a solution that fulfils the constraints, the term multidisciplinary consistency is used here.

The disciplines in aerospace MDO problems are generally linked by both shared and coupling variables. For example, the slender shapes of aeroplane wings result in structural deformations induced by the aerodynamic forces. These deformations in turn affect the aerodynamics of the structure and hence the aerodynamic forces. The structural and aerodynamic disciplines are thus coupled. Subspaces in MDO studies of automotive structures are usually linked by shared variables, but there are less common coupling variables that must be taken into account. Agte et al. (2010) refer to this as automotive designs are created in a multi-attribute environment rather than in a truly multidisciplinary one. This difference between aerospace and automotive MDO problems is interesting since many MDO methods are developed for aerospace applications. The question regarding the suitability of these methods for automotive structures then naturally arises.

Different approaches can be used to solve MDO problems. Methods suitable for large-scale problems aim at letting the groups involved work concurrently and autonomously. To let groups work concurrently increases efficiency as human and computational resources are used in parallel. Groups that work autonomously are not required to constantly share information with other groups. They can also govern the choice of methods and tools, and use their expertise to take part in design decisions. Multidisciplinary design optimization methods are either single-level or multi-level. The single-level methods have a central optimizer making all design decisions, while multi-level methods have a distributed optimization process. Multi-level methods were developed when MDO was applied to large problems in the aerospace industry involving several groups within a company. The intention was to distribute the work over many people and computers to compress the calendar time for problems with coupled disciplines (Kodiyalam and Sobieszczanski-Sobieski, 2001). However, multi-level methods complicate the solution process and to justify their use, the benefits must be greater than the cost.

3 Automotive Product Development

The development of a new car is a complicated task that requires many experts with different skills and responsibilities to cooperate in an organized manner. The product development process (PDP) describes what should be done at different stages of the development. It starts with initial concepts, which are gradually refined with the final aim of fulfilling all predefined targets. Many groups within the company organization are involved during the development. Some are responsible for designing a part of the product, e.g. the body, the interior, or the chassis system, while others are responsible for a performance aspect, e.g. crashworthiness, aerodynamics, or noise, vibration, and harshness (NVH). The groups work in parallel, and at certain times, the complete design is synchronized and evaluated. If the design is found to be satisfactory, the development is allowed to progress to the next phase.

Numerical simulations using finite element methods have been well integrated into the PDP for more than two decades, and more or less drive the development of today (Duddeck, 2008). Simulations can roughly be divided into two main categories in the same way as reflected by the groups within the organization of a company. The first one supports certain design areas and the other one evaluates disciplinary performance aspects that depends on more than one design area. The former is consequently evaluating many different aspects, e.g. stiffness, strength, and durability, for a certain area of the vehicle, while the latter focuses on one performance area, which often depends on the complete vehicle. One result of the increased focus on simulations is that the number of prototypes needed to test and improve different concepts has been reduced, although the amount of qualities to be considered during development has increased considerably. Hence, the extended use of simulations has resulted in both shortened development times and in reduced development costs. However, the increased demand of accuracy on the simulation models often results in detailed models that are time-consuming to evaluate. For example, it is not unusual that a crash model consists of several million elements and takes many hours to run on a high performance computing cluster.

To improve designs in a systematic way, different optimization methods have gained in popularity. Optimization can be used within different stages of the PDP: in the early phases to find promising concepts and in the later phases to fine-tune the design. Even if optimization has shown to result in better designs, the knowledge and additional resources

needed have delayed the use of its full potential. Optimization studies are often performed as an occasional effort when considered appropriate, and the time and scope are normally not defined in the PDP. This is certainly the case for MDO and it is therefore important to find methods that can fit into a modern PDP without jeopardizing its strict time limits. Metamodel-based design optimization is an approach that can make it possible to include also expensive simulation models in optimization studies.

3.1 Multidisciplinary Design Optimization of Structures

A typical MDO problem for automotive structures is to minimize the mass subject to a number of performance constraints originating from different disciplines. Other possibilities include finding the best compromise between conflicting requirements from different disciplines. In the simplest case, the appropriate thicknesses of selected parts are sought, but also the most suitable shape or material quality etc. can be found.

Multidisciplinary design optimization studies with full vehicle models are still rare in the automotive industry. Optimization studies with multiple loadcases within the same discipline and MDO studies of parts or subsystems are probably more common. However, one type of frequently reported full vehicle MDO study is to minimize the mass of the vehicle body considering noise, vibration and harshness and crashworthiness. This is a problem where the loadcases are coupled through shared variables but not through coupling variables. Crashworthiness simulations are computationally expensive. It is therefore only in recent years, after high performance computing systems and the possibility of parallel computing became available, that it has been feasible to include full vehicle crashworthiness simulations in MDO studies. Several examples of NVH and crashworthiness MDO studies using different approaches are documented in the literature, see e.g. Craig et al. (2002), Sobieszczanski-Sobieski et al. (2001), Yang et al. (2001), Kodiyalam et al. (2004), Hoppe et al. (2005), Duddeck (2008), and Sheldon et al. (2011). The aforementioned studies use single-level methods and cover both metamodel-based and direct optimization. It is shown that MDO is successful in finding better designs but it is not described how the methods can be implemented into the product development process.

4 Metamodel-Based Design Optimization

A metamodel is an approximation of a detailed simulation model, i.e. a model of a model. It is called metamodel-based design optimization (MBDO) when metamodels are used for the evaluations during the optimization process. There are several descriptions on MBDO, see for example Simpson et al. (2001), Queipo et al. (2005), Wang and Shan (2007), Forrester and Keane (2009), Stander et al. (2010), and Ryberg et al (2012).

A metamodels is a mathematical description created based on a dataset of input and the corresponding output from a detailed simulation model, see Figure 4.1. The mathematical description, i.e. metamodel type, suitable for the approximation could vary depending on the intended use or the underlying physics that the model should capture. Different datasets are appropriate for building different metamodels. The process of where to place the design points in the design space, i.e. the input settings for the dataset, is called design of experiments (DOE). Traditionally, the metamodels have been simple polynomials, but other metamodels that are better at capturing complex responses increase in popularity. The number of simulations needed to build a metamodel depends largely on the number of variables. Variable screening is therefore often used to identify the important variables in order to reduce the size of the problem and decrease the required number of detailed simulations. Since metamodels are approximations, it is important to know the accuracy of the models, i.e. how well the metamodels represent the detailed simulation model. This can be done by studying various error measures, which are obtained using different approaches.

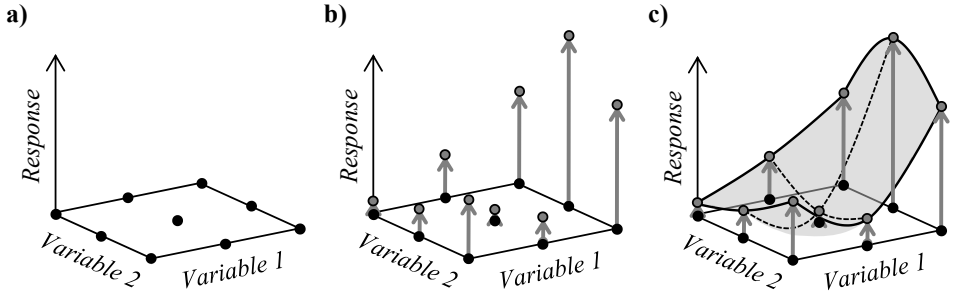

Figure 4.1 The concept of building a metamodel of a response depending on two design variables: **a)** design of experiments, **b)** function evaluations, and **c)** metamodel.

Metamodel-based design optimization can be performed using different strategies. One popular approach is the sequential response surface method. Simple polynomial (often linear) metamodels are then used for the evaluations during the optimization. The metamodels are built sequentially over a subregion of the design space, called region of interest, which is moved and reduced in size to close in on the optimum point. Another approach is to build metamodels that should capture the response over the complete design space. The size of the DOE is often gradually increased to achieve sufficiently accurate metamodels without spending too much computational effort. When the global metamodels are found to be adequately accurate, they are used for the evaluations during the optimization. This approach requires flexible metamodels that can adjust to an arbitrary number of points and capture complex responses.

Despite its simplicity, the sequential response surface method can work remarkably well and outperform the second approach if the global metamodels have insufficient accuracy, Duddeck (2008). However, many iterations can be required to find the optimum point for complex responses. The approach with global metamodels has the benefit of rendering a view of the complete design space. It is also suitable for finding Pareto optimal solutions during multi-objective optimization. Moreover, it is inexpensive to rerun optimizations, e.g. with changed constraint limits, once the global metamodels are built. One further benefit with this approach, when used in multidisciplinary design optimization, is the possibility for disciplinary autonomy. The different simulation experts can then be responsible for establishing the metamodels for their respective disciplines and loadcases, and for the validity of these metamodels. The development of the metamodels can be done in parallel, making the work efficient. Concurrency and autonomy are two of the main drivers for the various MDO multi-level optimization methods proposed, and the use of metamodels could thus have similar positive effects.

The MDO process proposed for automotive structures and presented in Chapter 6 is based on the second approach. The rest of this chapter therefore focuses on concepts related to global metamodels, suitable DOEs and optimization algorithms, as well as screening methods and metamodel validation that are relevant for such an approach.

4.1 Design of Experiments

A metamodel is build based on a dataset of input (design variable settings) and corresponding output (response values). The theory on where these design points should be placed in the design space in order to get the best possible information from a limited sample size is called design of experiments. The theories originate from planning physical experiments and focus on reducing the effect of noise. Popular designs include ***factorial*** or ***fractional factorial designs***, ***central composite designs***, ***Box-Behnken designs***, ***Plackett-Burman designs***, ***Koshal designs***, and ***D-optimal designs***, see e.g. Myers et al. (2008).

Classical DOEs tend to spread the sample points around the border and only put a few points in the interior of the design space. They are primarily used for screening purposes and to build polynomial metamodels. When the dataset is used to fit more advanced metamodels, other experimental designs are preferred. There seem to be a consensus among scientists that a proper experimental design for fitting global metamodels depending on many variables over a large design space should be space-filling. These types of DOEs aim at spreading the design points within the complete design space, which is desired when the form of the metamodel is unknown and when interesting phenomena can be found in any region of the design space. In addition to the different space-filling designs, different criteria-based designs can be constructed if certain information about the metamodel to be fitted is available a priori, which is not always the case. In an ***entropy design*** the purpose is to maximize the expected information gained from an experiment, while the ***mean squared error design*** minimizes the expected mean squared error (Koehler and Owen, 1996).

4.1.1 Latin Hypercube Sampling

The first space filling design, the ***Latin hypercube sampling*** (LHS), was proposed by McKay et al. (1979) and is a constrained random design. For each of the k variables the range of each variable is divided into n non-overlapping intervals of equal probability. One value from each interval is selected at random but with respect to the probability density in the interval. The n values of the first variable are then paired randomly with the n values of the second variable. These n pairs are combined randomly with the n values of the third variable to form n triplets, and so on, until n k-tuplets are formed, see Swiler and Wyss (2004) for a detailed description. This result in an $n \times k$ sampling plan matrix $\mathbf{S}$, where the k columns describe the levels of each variable, and the n rows describe the variable settings for each design, as shown in Figure 4.2. A common variant of LHS is the ***median Latin hypercube sampling*** (MLHS), which has points from the centre of the n intervals.

In order to generate a better space filling design, the LHS can be taken as a starting design and the values of each column in the sampling plan matrix permuted to optimize some criterion. One approach is to maximize the minimum distance between any two points, i.e. any two rows, with the help of an optimization algorithm. Another method is to minimize the discrepancy, which is a measure of non-uniformity of the points in the design space.

Orthogonal arrays (OAs) are highly fractionated factorial designs that also can be used to improve the LHS. One example is the ***randomized orthogonal array*** (Owen, 1992) in which the design space is divided into subspaces and not more than one design point is placed in each subspace. Another example is the ***orthogonal array-based Latin hypercubes*** (Tang, 1993), which is an LHS with the design space divided into subspaces

and not more than one design point placed in each subspace. A comparison between the LHS and these improved designs is found in Figure 4.3.

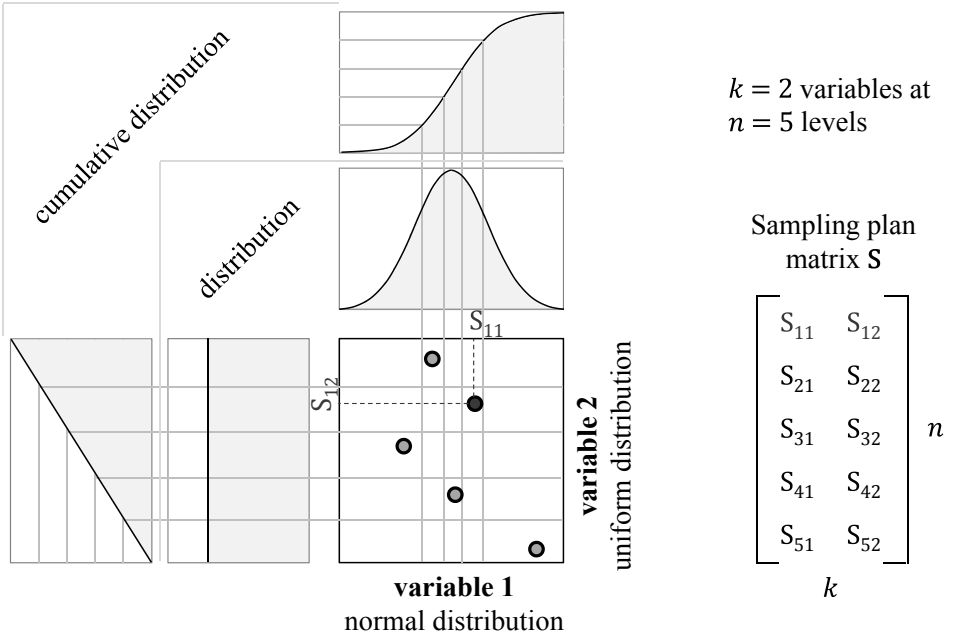

Figure 4.2 Latin hypercube sampling for two variables at five levels, one normally distributed variable and the other uniformly distributed.

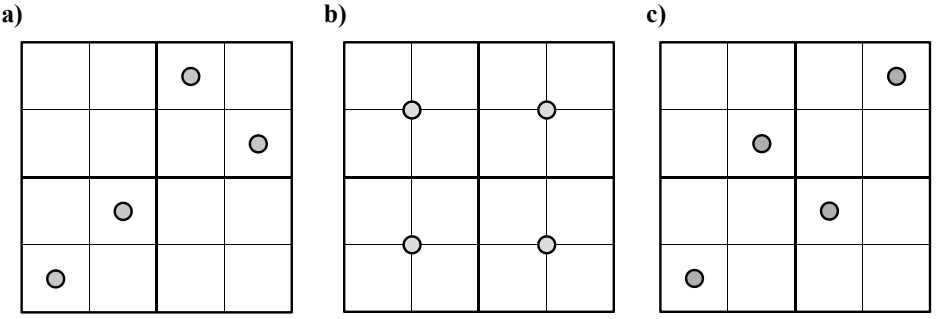

Figure 4.3 Comparison between different space-filling DOEs with two variables and four design points: **a)** Median Latin hypercube sampling, **b)** Randomized orthogonal array, and **c)** Orthogonal array-based Latin hypercube sampling.

4.1.2 Distance-Based Designs

In addition to the various LHS methods, several other space-filling methods exist. When n points are chosen within the design space so that the minimum distance between them are maximized, a ***maximin*** or sphere-packing design is obtained (Johnson et al., 1990). For small n this will generally result in the points lying on the exterior of the design space and that the interior is filled as the number of points becomes larger. Another of the so-called distance-based designs is the ***minimax design***, where the maximum distance between any design points is minimized. In this case, the designs will generally lie in the interior of the design space also for small numbers of n, as can be observed from Figure 4.4.

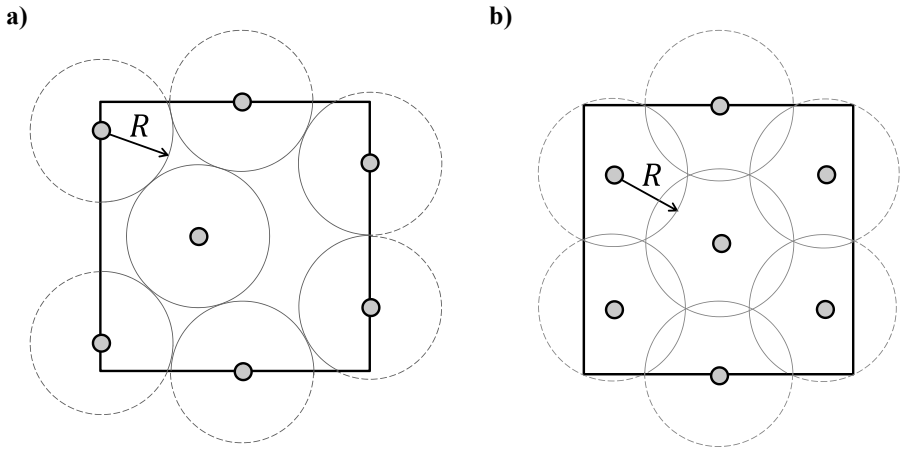

Figure 4.4 Comparison of maximin and minimax designs with seven points in two variables. **a)** Maximin, where the design space is filled with spheres with maximum radius **b)** Minimax, where the design space is covered by spheres with minimum radius.

4.1.3 Low-Discrepancy Sequences

Hammersley sequence sampling (HSS) (Kalagnanam and Diwekar, 1997) and ***uniform designs*** (UD) (Fang et al., 2000) belong to a group called low-discrepancy sequences. The discrepancy is a measure of the deviation from a uniform distribution and could be measured in several ways. While LHS is uniform only in a one-dimensional projection, these methods tend to be more uniform in the entire design space. In HSS, the low discrepancy sequence of Hammersley points is used to sample the k-dimensional space. The UD, on the other hand, has similarities with LHS. In the UD, the points are always selected from the centre of cells in the same way as for the MLHS. In addition to the one-dimensional balance of all levels for each factor in the LHS, the UD also requires k-dimensional uniformity. The most popular UD, the U UD, could be obtained by selecting the design with the smallest discrepancy out of all possible MLHS designs.

4.1.4 Sampling Size and Sequential Sampling

Several factors are important for determining how well the metamodel will fit the true response. Two of the important factors are the number of design points used for fitting the model and their distribution in the design space. In order to build a polynomial metamodel, there is a fixed minimum number of design points required, depending on the number of variables. However, it is usually desirable to use a larger sampling size than the minimum required, i.e. to use oversampling, to improve the accuracy and have the possibility to estimate how good the metamodel is. For many of the more advanced metamodels, there is no such minimum sample size, although the accuracy of the metamodel will be limited if the sampling size is too small. Also, the more complex response the metamodel should capture, the larger sample size it requires.

Detailed simulation models are often time-consuming to run. The question in practice is therefore often how many design points that are needed to fit a reasonably accurate metamodel. It has been proposed by Gu and Yang (2006) and Shi et al. (2012) that a minimum of $3k$ sampling points, where k equals the number of variables, are needed to build a reasonably accurate metamodel. An initial sampling size of between $3k$ and $4k$ could therefore be sensible, at least if k is not too large. It is, however, difficult to know the appropriate sampling size beforehand. Therefore sequential sampling can be used to avoid issues with too many, i.e. unnecessary time-consuming, or too few design points, resulting in low metamodel accuracy. A limited number of designs could thus be used as a starting point and, if required, additional points could be added later. It has been shown by Jin et al. (2002) that the performance of sequential sampling approaches generally is comparable to selecting all points at once.

Many different sequential approaches have been proposed (Jin et al., 2002; Forrester and Keane, 2009) and they are typically based on some optimality criteria. When information from previously fitted metamodels is used in the sequential sampling, the sampling is said to be adaptive. However, not all models provide the necessary estimation of the prediction error directly and cross validation can then be used for this estimation (see Section 4.4.2). A common alternative sequential sampling technique, which is not adaptive, is the maximin distance approach. Given an existing sample set, the idea is to select the new sample set so that the minimum distance between any two points in the complete set is maximized.

4.2 Screening

The number of simulations needed to build a metamodel depends on the number of design variables. Eliminating the variables that do not influence the results can therefore substantially reduce the computational cost. The process of studying the importance of different

variables, identifying the ones to be included, and eliminating the ones that do not influence the responses is called variable screening.

Several screening methods exist, see e.g. Viana et al. (2010). One of the simplest screening techniques uses one-factor-at-a-time plans, which evaluate the effect of changing one variable at a time. It is a very inexpensive approach but it does not estimate the interaction effects between variables. Therefore, variants of this method that account for interactions have been proposed. One example is ***Morris method*** (Morris, 1991) which, at the cost of additional runs, tries to determine whether the variables have effects that are (a) negligible, (b) linear and additive, or (c) non-linear or involved in interactions with other variables.

Another category of screening techniques are variance-based. One simple and commonly used approach is based on ***analysis of variance*** (ANOVA) as described by Myers et al. (2008). The idea is to fit a metamodel using regression analysis, e.g. a simple polynomial metamodel, and study the coefficients for each term in the model. The importance of a variable can then be judged both by the magnitude of the related estimated regression coefficients and by the level of confidence that the regression coefficient is non-zero. This technique is used to separately identify the main and interaction effects that account for most of the variance in the response.

An alternative variance-based method is Sobol's ***global sensitivity analysis*** (GSA), which provides the total effect (main and interaction effects) of each variable (Sobol', 2001). The method can be used for arbitrary complex metamodels and includes the calculation of sensitivity indices. These indices can be used to rank the importance of the design variables for a response and thus identify insignificant design variables. It is also possible to quantify what amount of the variance that is caused by a single variable.

4.3 Metamodels

When running a detailed simulation model, a vector of input (design variable values) results in a vector of output (response values). Each element in the response vector represents a specific response. For each of these responses, a metamodel can be built to approximate the true response. The metamodel is built from a dataset of input design points $\mathbf{x}_i = (x_1, x_2, \ldots, x_k)^T$ and the corresponding output responses $y_i = f(\mathbf{x}_i)$, where k is the number of design variables, $i = 1, \ldots, n$, and n is the number of designs used to fit the model. For an arbitrary design point $\mathbf{x}$, the predicted response $\hat{y}$ will differ from the true response y of the detailed model, i.e.

$$y = f(\mathbf{x}) = \hat{y} + \varepsilon = s(\mathbf{x}) + \varepsilon \tag{4.1}$$

Here, $f(\mathbf{x})$ represents the detailed model, $s(\mathbf{x})$ is the mathematical function defining the

metamodel, and ε is the approximation error. Several mathematical formulations can be used for the metamodels. They all have their unique properties and there is no universal model that always is the best choice. Instead, the suitable metamodel depends on the problem at hand. It is for example important to decide whether the model should be a global or a local approximation. A basic knowledge about the complexity of the response the metamodel should capture is useful when choosing between metamodel types. Another issue that needs to be considered is whether or not noise is present in the fitting set. An interpolating model might be the best choice in the noise-free case, while an approximating model may be better when noise is present. However, it should be noted that there is no guarantee that an interpolating model produces better predictions in unknown points compared to an approximating one, even if there is no noise present.

Many comparative studies have been made over the years to guide the selection of metamodel types, see e.g. Jin et al. (2001), Clarke et al. (2005), Kim et al. (2009), and Li et al. (2010). Despite this, it is not possible to draw any decisive conclusions regarding the superiority of any of the metamodel types. In addition, there are often several parameters that must be tuned when a metamodel is built. This means that results can differ considerably depending on how well these parameters are tuned and, consequently, the results also depend on the software used to build the metamodel.

Instead of selecting only the assumed best metamodel, several different metamodels can be combined. The idea is that the combined model should perform at least as well as the best individual metamodel, but at the same time protect against the worst individual metamodel. A ***weighted average surrogate*** (WAS) makes a weighted linear combination of metamodels in the hope of cancelling out prediction errors through a proper selection of the weights. A metamodel that is judged to be more accurate should be assigned a large weight, and a less accurate metamodel should be assigned a lower weight resulting in a smaller influence on the predictions. The evaluation of the accuracy is done with different measures of goodness of fit and could be either global or local. When weights are selected based on a global measure, the weights are fixed (Goel et al., 2007) and when the weights are based on a local measure, the weights are instead functions of space (Zerpa et al., 2005). In the latter case, different metamodels can have the largest influence on the prediction in different areas of the design space.

Another way of combining metamodels can be used if enough samples exist in the fitting set. A ***multi-surrogate approximation*** (MSA) is created by first classifying the given samples into clusters based on their similarities in the design space. Then, a proper local metamodel is identified for each cluster and a global metamodel is constructed using these local metamodels (Zhao and Xue, 2011). This method is particularly useful when sample data from various regions of the design space are of different characteristics, e.g. with and without noise.

Traditionally, ***polynomial metamodels*** have often been used. These models are developed using regression, i.e. fitting a regression model $y = s(\mathbf{x},\boldsymbol{\beta}) + \varepsilon$ to a dataset of n variable settings $\mathbf{x}_i$ and corresponding responses y_i. The method of least squares chooses the regression coefficients $\boldsymbol{\beta}$ so that the quadratic error is minimized. The least square estimators of the regression coefficients are denoted $\mathbf{b}$ and can be found using matrix algebra (Myers et al., 2008) as

$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{4.2}$$

where $\mathbf{y}$ is the vector of n responses used to fit the model depending on k variables and $\mathbf{X}$ is the model matrix

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} & x_{11}x_{12} & \cdots & x_{1(k-1)}x_{1k} & x_{11}^2 & \cdots & x_{1k}^2 & \cdots \\ 1 & x_{21} & \cdots & x_{2k} & x_{21}x_{22} & \cdots & x_{2(k-1)}x_{2k} & x_{21}^2 & \cdots & x_{2k}^2 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \cdots \\ 1 & x_{n1} & \cdots & x_{nk} & x_{n1}x_{n2} & \cdots & x_{n(k-1)}x_{nk} & x_{n1}^2 & \cdots & x_{nk}^2 & \cdots \end{bmatrix} = \begin{bmatrix} \mathbf{f}^T(\mathbf{x}_1) \\ \mathbf{f}^T(\mathbf{x}_2) \\ \vdots \\ \mathbf{f}^T(\mathbf{x}_n) \end{bmatrix} \tag{4.3}$$

In this matrix, each row corresponds to one fitting point and each column is related to one regression coefficient, i.e. the number of columns depends on the polynomial order and how many interactions that are considered. The resulting polynomial metamodel becomes

$$\hat{y}(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})\mathbf{b} \tag{4.4}$$

This metamodel will in general not interpolate the fitting data. One exception is when the fitting set is so small that there is just enough data to determine all the regression coefficients. However, such small fitting sets are generally not recommended. Low order polynomial metamodels will capture the global trends of the detailed simulation model, but will in many cases not be a good representation of the complete design space. These metamodels are therefore mainly used for screening purposes and in iterative optimization procedures.

Polynomial metamodels can produce large errors for highly non-linear responses but can provide good local approximations if the response is less complex. These features are taken advantage of in the method of ***moving least squares*** (MLS). For a specific value of $\mathbf{x}$, a polynomial is fitted according to the least squares method, but the influence of surrounding points is weighted depending on the distance to $\mathbf{x}$ (Breitkopf et al., 2005). Hence, compared to Equation (4.4) for polynomial metamodels, the MLS model has coefficients $\mathbf{b}$ that depend on the location in the design space, i.e. depend on $\mathbf{x}$. Thus, one polynomial fit is not valid over the entire domain as for normal polynomial metamodels. Instead, the polynomial is valid only locally around the point $\mathbf{x}$ where the fit is made. Since $\mathbf{b}$ is a function of $\mathbf{x}$, a new MLS model needs to be fitted for each new evaluation. Furthermore, in order to construct the metamodel, a certain number of fitting points must fall within the domain of influence. The number of influencing fitting designs can be

adjusted by changing the weight functions, or rather the radius of the domain of influence. The denser the design space is sampled, the smaller the domain of influence can be, and the more accurate the metamodel becomes.

Next, some other metamodels suitable for global approximations and frequently mentioned in the literature will be covered in more detail. These metamodels could be possible alternatives for the MDO process presented in Chapter 6.

4.3.1 Kriging

Kriging is named after D. C. Krige, and this method for building metamodels has been used in many engineering applications. Design and analysis of computer experiments (DACE) is a statistical framework for dealing with Kriging approximations to complex and expensive computer models presented by Sacks et al. (1989). The idea behind Kriging is that the deterministic response $y(\mathbf{x})$ can be described as

$$y(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x}) \tag{4.5}$$

where $f(\mathbf{x})$ is a known polynomial function of the design variables $\mathbf{x}$ and $Z(\mathbf{x})$ is a stochastic process (random function). This process is assumed to have mean zero, variance σ^2 and a non-zero covariance. The $f(\mathbf{x})$ term is similar to a polynomial model described in the previous section and provides a global model of the design space, while the $Z(\mathbf{x})$ term creates local deviations so that the Kriging model interpolates the n sampled data points. In many cases, $f(\mathbf{x})$ is simply a constant term and the method is then called ***ordinary Kriging***. If $f(\mathbf{x})$ is set to 0, implying that the response $y(\mathbf{x})$ has mean zero, the method is called ***simple Kriging***.

A fitted Kriging model for an unknown point $\mathbf{x}$ can be written as

$$\hat{y}(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})\mathbf{b} + \mathbf{r}^T(\mathbf{x})\mathbf{R}^{-1}(\mathbf{y} - \mathbf{X}\mathbf{b}) \tag{4.6}$$

where $\mathbf{f}(\mathbf{x})$ is a vector corresponding to a row of the model matrix $\mathbf{X}$ in the same way as for the polynomial models previously described. $\mathbf{b}$ is a vector of the estimated regression coefficients, $\mathbf{r}(\mathbf{x}) = [R(\mathbf{x}, \mathbf{x}_1), R(\mathbf{x}, \mathbf{x}_2), \ldots, R(\mathbf{x}, \mathbf{x}_n)]^T$ is a vector of correlation functions between the unknown point and the n sample points, $\mathbf{R}$ is the matrix of correlation functions for the fitting sample, and $\mathbf{y}$ is a vector of the observed responses in the fitting sample. The term ($\mathbf{y}$ - $\mathbf{X}\mathbf{b}$) is a vector of residuals for all fitting points when the stochastic term of the model is disregarded. The regression coefficients are found by

$$\mathbf{b} = (\mathbf{X}^T\mathbf{R}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{R}^{-1}\mathbf{y} \tag{4.7}$$

Many different correlation functions could be used, but two commonly applied functions are the exponential and the Gaussian correlation functions (Stander et al., 2010), i.e.

$$R(\mathbf{x}_i, \mathbf{x}_j) = \prod_{r=1}^{k} e^{-\theta_r \left| x_i^r - x_j^r \right|} \tag{4.8}$$

and

$$R(\mathbf{x}_i, \mathbf{x}_j) = \prod_{r=1}^{k} e^{-\theta_r \left| x_i^r - x_j^r \right|^2} \tag{4.9}$$

respectively. $|x_i^r - x_j^r|$ is the distance between the i^{th} and j^{th} sample point of variable x^r, k is the number of variables, and θ_r is the correlation parameter for variable x^r. In general, a different θ_r for each variable is used, which yields a vector $\boldsymbol{\theta}$ with k elements. In some cases, a single correlation parameter for all variables produces sufficiently good results, and the model is then said to be isotropic. The parameter θ_r is essentially a width parameter that affects how far the influence of a sample point extends (Forrester and Keane, 2009). A low θ_r means that all points will have a high correlation R, with $Z(x^r)$ being similar across the sample, while a high θ_r means that there is a significant difference between the $Z(x^r)$ for different sample points. The elements of $\boldsymbol{\theta}$ could therefore be used to identify the most important variables, provided that a suitable scaling of the design variables is used.

In order to build a Kriging metamodel, the correlation parameters $\boldsymbol{\theta}$ must be determined. The optimum values of $\boldsymbol{\theta}$ can be found by solving the non-linear optimization problem of maximizing the log-likelihood function

$$\begin{aligned} &\max_{\boldsymbol{\theta}} && L(\boldsymbol{\theta}) = -\frac{1}{2}[n \ln(\hat{\sigma}^2) + \ln|\mathbf{R}|] \\ &\text{subject to} && \theta_r > 0, r = 1, \dots, k \end{aligned} \tag{4.10}$$

where $|\mathbf{R}|$ is the determinant of $\mathbf{R}$ and the estimate of the variance is given by

$$\hat{\sigma}^2 = \frac{(\mathbf{y} - \mathbf{X}\mathbf{b})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{X}\mathbf{b})}{n} \tag{4.11}$$

An equivalent problem to problem (4.10) is to minimize $\hat{\sigma}^2 |\mathbf{R}|^{(1/n)}$ for $\boldsymbol{\theta} > 0$. These are k-dimensional optimization problems that can require significant computational time to solve if the fitting set is large. Additionally, the correlation matrix can become singular if the sample points are too close to each other, or if the sample points are generated from particular DOEs. A small adjustment of the $\mathbf{R}$-matrix can avoid ill-conditioning but might result in a metamodel that does not interpolate the observed responses exactly.

When working with noisy data, an interpolating model might not be desirable. Special choices of correlation functions can then result in metamodels that approximate the fitting

data (Simpson et al., 2001). An interpolating Kriging model can also be modified by adding a regularization constant to the diagonal of the correlation matrix so that the model does not interpolate the data. The Kriging method is thus flexible and well suited for global approximations of the complete design space. Kriging models also provide an estimate of the prediction error in an unobserved point directly (Sacks et al., 1989), which is a feature that can be used in adaptive sequential sampling approaches.

4.3.2 Radial Basis Functions

Radial basis function (RBF) methods for interpolating scattered multivariate (multiple variables) data were first studied by R. Hardy and a description could be found in Hardy (1990). Radial basis functions depend only on the radial distance from a specific point $\mathbf{x}_i$ such that

$$\phi(\mathbf{x}, \mathbf{x}_i) = \phi(\|\mathbf{x} - \mathbf{x}_i\|) = \phi(r) \tag{4.12}$$

where r is the distance between the points $\mathbf{x}$ and $\mathbf{x}_i$. The RBFs can be of many forms but are always radially symmetric. The Gaussian function and Hardy's multiquadrics are commonly used and expressed as

$$\phi(r) = e^{-\left(\frac{r}{c}\right)^2} \tag{4.13}$$

and

$$\phi(r) = \sqrt{r^2 + c^2} \tag{4.14}$$

respectively, where c is a shape parameter that controls the smoothness of the function, see Figure 4.5.

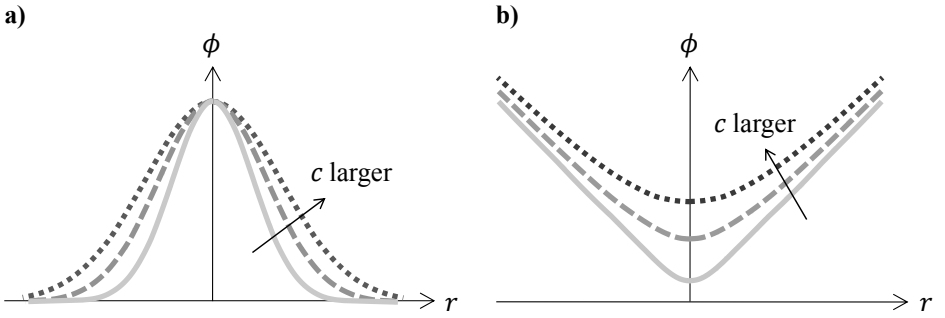

Figure 4.5 Examples of radial basis functions: **a)** Gaussian RBF and **b)** Hardy's multi-quadric RBF.

An RBF metamodel consists of a linear combination of radially symmetric functions to approximate complex responses and can be expressed as

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{n} w_i \, \phi(\|\mathbf{x} - \mathbf{x}_i\|) = \mathbf{w}^T \boldsymbol{\phi} \tag{4.15}$$

The metamodel is thus represented by a sum of n RBFs, each associated with a sample point $\mathbf{x}_i$, representing the centre of the RBF, and weighted by a coefficient w_i. The coefficients w_i, i.e. the unknown parameters that must be determined when building the metamodel, can be collected in a vector $\mathbf{w}$. The vector $\boldsymbol{\Phi}$ contains the evaluations of the RBF for all distances between the studied point $\mathbf{x}$ and the sample designs $\mathbf{x}_i$.

Radial basis function metamodels are often interpolating, i.e. the parameters w_i are chosen such that the approximation matches the responses in the sampled dataset $(\mathbf{x}_i, y_i)$, where $i = 1, \ldots, n$. This can be obtained if the number of RBFs equals the number of samples in the fitting set, resulting in a linear system of equations in w_i

$$\mathbf{y} = \mathbf{B}\mathbf{w} \tag{4.16}$$

where $\mathbf{y}$ is the vector of responses, $\mathbf{w}$ is the vector of unknown coefficients, and $\mathbf{B}$ is the $n \times n$ symmetric interpolation matrix that contain evaluations of the RBF for the distances between all the fitting points

$$B_{ij} = \phi\left(\left\|\mathbf{x}_i - \mathbf{x}_j\right\|\right) \tag{4.17}$$

The equation system (4.16) can be solved by standard methods, using matrix decompositions, for small n, but special methods have to be applied when n becomes large (Dyn et al., 1986), since the interpolation matrix is often full and ill-conditioned.

When the number of basis functions n_{RBF} is smaller than the sample size n_s, the model will be approximating. Similarly to the polynomial regression model, the optimal weights in the least squares sense is obtained as

$$\mathbf{w} = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{y} \tag{4.18}$$

where $\mathbf{B}$ is an $n_s \times n_{RBF}$ matrix with elements B_{ij} as described in Equation (4.17) for $i = 1, \ldots, n_s$ and $j = 1, \ldots, n_{RBF}$, and $\mathbf{x}_j$ represents the centre of the basis functions.

The shape parameter c in Equations (4.13) and (4.14) plays an important role since it affects the conditioning of the problem. When $c \to \infty$, the elements of the interpolation matrix $\mathbf{B}$ approach constant values and the problem becomes ill-conditioned. In a physical sense, the shape parameter c controls the width of the functions and thereby the influence of nearby points. A large value of c gives a wider affected region, i.e. points further away from an unknown point will have an effect on the prediction of the response at the un-

known point. A small value of c, on the other hand, means that only nearby points will influence the prediction. Consequently, the selection of c also influences the risk of overfitting or underfitting. If the value is chosen too small, overfitting will occur, i.e. every sample point will influence only the very close neighbourhood. On the other hand, if the value is selected too large, underfitting will appear and the model loses fine details, see Figure 4.6. So, while the correct choice of **w** will ensure that the metamodel can reproduce the training data, the correct estimate of c will enable a smaller prediction error in unknown points. The prediction error for a RBF metamodel can easily be evaluated at any point in the design space (Forrester and Keane, 2009), which is a property that can be useful in e.g. sequential sampling.

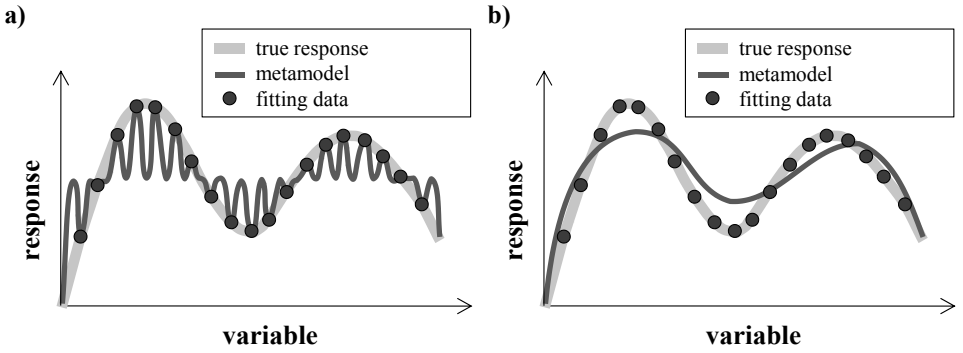

Figure 4.6 Examples of models with poor prediction capabilities due to **a)** overfitting and **b)** underfitting.

4.3.3 Artificial Neural Networks

Artificial neural networks are intended to respond to stimulus in a fashion similar to the biological nervous systems. One of the attractive features of these structures is their ability to learn associations between data. An artificial neural network, or often just neural network (NN), may therefore be used to approximate complex relations between a set of input and output, and can thus serve as a metamodel.

An NN is composed of small computing elements called neurons, assembled into an architecture. Based on the input $\mathbf{x} = (x_1, x_2, \ldots, x_k)^T$, the output y_m from a single neuron m is evaluated as

$$y_m(\mathbf{x}) = f\left(b_m + \sum_{i=1}^{k} w_{mi} x_i\right) = f(a) \tag{4.19}$$

where f is the transfer or activation function, b_m is the bias value, and w_{mi} the weight of

the corresponding input x_i for neuron m. A schematic description is presented in Figure 4.7. The input **x** to the neuron are either variable values or output from previous neurons in the network. The connection topology of the architecture, the weights, the bias, and the transfer function used determine the form of the neural network.

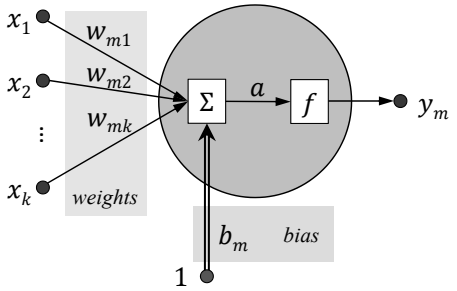

Figure 4.7 Illustration of neuron m in a neural network, where input is variables or output from previous neurons.

One very common architecture is the ***multi-layer feedforward neural network*** (FFNN), see Figure 4.8, in which the information only is passed forward in the network and no information is fed backward. The transfer function in the hidden layers of an FFNN is often a sigmoid function, i.e.

$$f(a) = \frac{1}{1 + e^{-a}} \tag{4.20}$$

which is an S-shaped curve ranging from 0 to 1 and a is defined in Equation (4.19). For the input and output layers, a linear transfer $f(a) = a$ is often used with bias added to the output layer but not to the input layer. This means that a simple neural network with only one hidden layer of M neurons can be of the form

$$\hat{y}(\mathbf{x}) = b + \sum_{m=1}^{M} \frac{w_m}{1 + e^{-(b_m + \sum_{i=1}^{k} w_{mi} x_i)}} \tag{4.21}$$

where b is the bias of the output neuron, w_m is the weight on the connection between the m^{th} hidden neuron and the output neuron, b_m is the bias in the m^{th} hidden neuron, and w_{mi} is the weight on the connection between the i^{th} input and the m^{th} hidden neuron.

There are two distinct steps in building a neural network. The first is to choose the architecture and the second is to train the network to perform well with respect to the training set of input (design variable values) and corresponding output (response values). The second step means that the free parameters of the network, i.e. the weights and biases

in the case of an FFNN, are determined. This is a non-linear optimization problem in which some error measure is minimized.

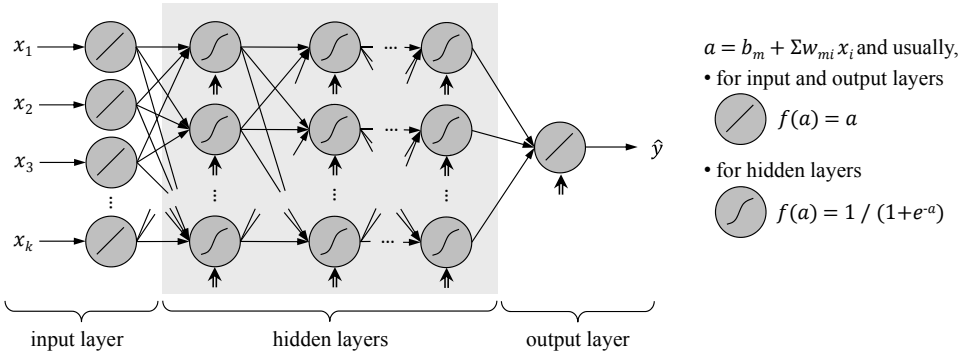

Figure 4.8 Illustration of a feedforward neural network architecture with multiple hidden layers**.**

If the steepest descent algorithm is used for the optimization, the training is said to be done by back-propagation (Rumelhart et al., 1986), which means that the weights are adjusted in proportion to

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w_{ji}} \tag{4.22}$$

The studied error measure E is the sum of the squared differences between the target output and the actual output from the network over all n points in the training set, i.e.

$$E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4.23}$$

The adjustment of the weights starts at the output layer and is thus based on the difference between the response from the NN and the target response from the training set. For the hidden layers, where there is no specified target value y_i, the adjustments of the weights are instead determined recursively based on the sum of the changes at the connecting nodes multiplied with their respective weights. In this way the adjustments of the weights are distributed backwards in the network, and hence the name back-propagation.

It has been shown by Hornik et al. (1989) that FFNNs with one hidden layer can approximate any continuous function to any desired degree of accuracy, given a sufficient number of neurons in the hidden layer and the correct interconnection weights and biases. In

theory, FFNN metamodels thus have the flexibility to approximate very complex functions, and FFNNs are therefore well suited for global approximations of the design space.

The decision of the appropriate number of neurons in the hidden layer or layers is not trivial. Generally, the correct number of neurons in the hidden layer(s) is determined experimentally, i.e. a number of candidate networks are constructed and the one judged to be the best is selected. Only one hidden layer is often used. Although FFNNs with one hidden layer theoretically should be able to approximate any continuous function, only one hidden layer is not necessarily optimal. One hidden layer may require many more neurons to accurately capture complex functions than a network with two hidden layers, since it might be easier to improve an approximation locally without making it worse elsewhere in a network with two hidden layers (Chester, 1990).

Evidently, if the number of free parameters is sufficiently large and the training optimization is run long enough, it is possible to drive the training error as close to zero as preferred. However, a too small error is not desirable since it can lead to overfitting instead of a model with good prediction capabilities. An overfitted model does not capture the underlying function properly. It describes the noise rather than the principal relationship and can result in poor predictions even for noise-free data, see Figure 4.9a. Overfitting generally occurs when a model is excessively complex, i.e. when it has too many parameters relative to the number of observations in the training set. On the other hand, if the network model is not sufficiently complex, the model can also fail in capturing the underlying function, leading to underfitting, see Figure 4.9b. Given a fixed amount of training data, it is beneficial to reduce the number of weights and biases as well as the size of them in order to avoid overfitting.

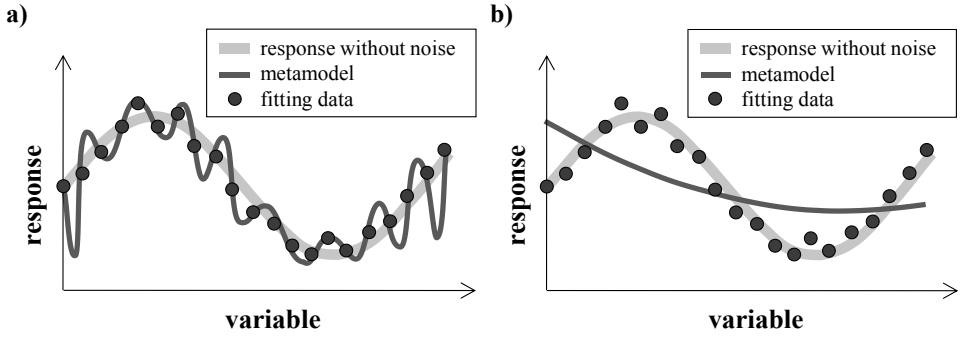

Figure 4.9 Examples of models with poor prediction capabilities due to **a)** overfitting and **b)** underfitting.

Regularization means that some constraints are applied to the construction of the NN model in order to reduce the prediction error in the final model. For FFNN models,

regularization can be done by controlling the number of hidden neurons in the network. Another way is to impose penalties on the weights and biases or to use a combination of both methods (Stander et al., 2010). A fundamental problem when modelling noisy or using very limited data is to balance between the goodness of fit and the choice of how strong the constraints forced on the model by regularization should be.

Another common type of neural network, in addition to the FFNN, is the ***radial basis function neural network*** (RBFNN), which has activation functions in the form of RBFs. An RBFNN has a defined three-layer architecture with the single hidden layer built of non-linear radial units, each responding only to a local region of the design space. The input layer is linear and the output layer performs a biased weighted sum of the hidden layer units and creates an approximation over the entire design space, see Figure 4.10. The RBFNN model is sometimes complemented with a linear part corresponding to additional direct connections from the input neurons to the output neuron.

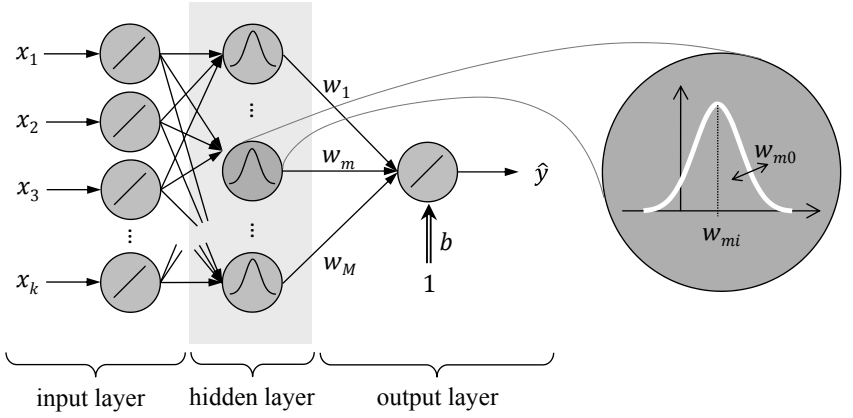

Figure 4.10 Illustration of a radial basis function neural network with Gaussian activation functions.

Gaussian functions and Hardy's multiquadrics, respectively, as defined in Equations (4.13) and (4.14), are commonly used RBFs. The activation of the m^{th} RBF is determined by the Euclidean distance

$$r = \sqrt{\sum_{i=1}^{k} (x_i - w_{mi})^2} \tag{4.24}$$

between the input vector $\mathbf{x} = (x_1, \ldots, x_k)^T$ and the RBF centres $\mathbf{w}_m = (w_{m1}, \ldots, w_{mk})$ in the k-dimensional space. For a given input vector $\mathbf{x}$, the output from an RBFNN with k input

neurons and a hidden layer consisting of M RBF units (but without a linear part) is given by

$$\hat{y} = b + \sum_{m=1}^{M} \mathrm{w}_m f(a_m) \tag{4.25}$$

where, in the case of a Gaussian model,

$$f(a_m) = e^{-(\frac{r}{c})^2} = e^{-w_{m0} \sum_{i=1}^{k} (x_i - w_{mi})^2} \tag{4.26}$$

Thus, the hidden layer parameters $\mathbf{w}_m = (w_{m1}, \ldots, w_{mk})$ represent the centre of the m^{th} radial unit, while w_{m0} determines its width. The parameters b and $w_1, \ldots, w_M$ are the bias and weights of the output layer, respectively. All these parameters and the number of neurons M must be determined when building the RBFNN metamodel.

In the same way as a feedforward neural network can approximate any continuous function to any desired degree of accuracy, an RBFNN with enough hidden neurons can too. An important feature of the RBFNNs, which differs from the FFNNs, is that the hidden layer parameters, i.e. the parameters governing the RBFs, can be determined by semi-empirical, unsupervised training techniques. This means that RBFNNs can be trained much faster than FFNNs although the RBFNN may require more hidden neurons than a comparable FFNN (Stander et al., 2010).

The training process for RBFNNs are generally done in two steps. First, the hidden layer parameters, i.e. the centre and width of the radial units, are set. Then, the bias and weights of the linear output layer are optimized, while the basis functions are kept fixed. In comparison, all of the parameters of an FFNN are usually determined simultaneously as part of a single optimization procedure (training). The optimization in the second step of the RBFNN training is done to minimize some performance criterion, e.g. the mean sum of squares of the network errors on the training set (MSE), see Equation (4.39). If the hidden layer parameters are kept fixed, the MSE performance function is a quadratic function of the output layer parameters and its minimum can be found as the solution to a set of linear equations. The possibility to avoid time consuming non-linear optimization during the training is one of the major advantages of RBFNNs compared to FFNNs.

Commonly, the number of RBFs are chosen to be equal to the number of samples in the training dataset ($M = n$), the RBF centres are set at the fitting designs ($\mathbf{w}_m = \mathbf{x}_m$, $m = 1, \ldots, n$), and the widths of the radial units are all selected equal. In general, the widths are set to be a multiple s_w of the average distance between the RBF centres so that they overlap to some degree and hence result in a relatively smooth representation of the data. Sometimes the widths are instead individually set to the distance to the n_w ($<< n$) closest neighbours so that the widths become smaller in areas with many samples close to

each other. This results in a model that preserves fine details in densely populated areas and interpolates the data in sparse areas of the design space.

When building an RBFNN metamodel, the goal is to find a smooth model that captures the underlying functional response without fitting potential noise, i.e. avoid overfitting. For noisy data, the exact RBFNN that interpolates the training dataset is usually a highly oscillatory function, and this needs to be addressed when building the model. Similarly as can be done for an FFNN or a Kriging model, regularization can be applied to adjust the output layer parameters in the second phase of the training. This will yield a model that no longer passes through the fitting points. However, it is probably more effective to properly select the hidden layer parameters, i.e. the width and centres of the RBF units, in the first step of the training. Regularization in the second step can never compensate for large inaccuracies in the model parameters. Another way of constructing an approximating model is to reduce the number of RBFs. This could be done by starting with an empty subset of basis functions and adding, one at a time, the basis function which reduces some error metric the most. The selection is done from the n possible basis functions, which are centred around the observed data points $\mathbf{x}_i$, and the process is continued until no significant decrease in the studied error metric is observed.

Since the accuracy of the metamodel strongly depends on the hidden layer parameters, it is important to estimate them well. Instead of just selecting the values, the widths can be found by looping over several trial values of s_w or n_w and finally selecting the best RBFNN. The selection can for example be based on the generalized cross validation error, which is a measure of goodness of fit that also takes the model complexity into account, see Section 4.4.3. Another approach to find the best possible RBFNN metamodel can be to include the widths as adjustable parameters along with the output layer parameters in the second step of training. However, this requires a non-linear optimization in combination with a sophisticated regularization, and one of the benefits with the RBFNN, the speed of training, will be lost.

4.3.4 Multivariate Adaptive Regression Splines

Multivariate adaptive regression splines (MARS) is a regression procedure that automatically models non-linearities and interactions but normally not interpolate the fitting data (Friedman, 1991). The approximation does not have a predefined form but is constructed based on information derived from the fitting data. The MARS procedure builds the metamodel from a set of coefficients a_m and basis functions B_m that are adaptively selected through a forward and backward iterative approach.

A MARS model is built from truncated power functions representing q^{th} order splines

$$b_q^+(x-t) = [+(x-t)]_+^q = (\max\{0,(x-t)\})^q$$
$$b_q^-(x-t) = [-(x-t)]_+^q = (\max\{0,-(x-t)\})^q = (\max\{0,(t-x)\})^q \quad (4.27)$$

where t is the truncation location, also called knot location, and q is the order of the spline. The subscript "+" indicates that the argument, i.e. the value within the squared brackets, should be positive, otherwise is the function assumed to be zero. For $q > 0$, the spline is continuous and has q - 1 continuous derivatives. Often $q = 1$ is recommended and the splines then become "hinge functions", see Figure 4.11. The resulting MARS model will then have discontinuous derivatives but could be modified to have continuous first order derivatives (Friedman, 1991).

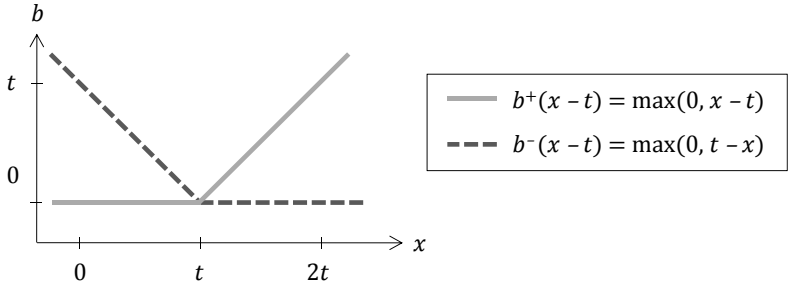

Figure 4.11 A mirrored pair of hinge functions with the knot at $x = t$.

A MARS metamodel can be written as

$$\hat{y}(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m B_m(\mathbf{x}) \quad (4.28)$$

which could be seen as a weighted sum of basis functions.

The coefficients a_m are estimated through least-squares regression of the basis functions $B_m(\mathbf{x})$ to the responses y_i (i =1, ... , n) in the fitting set. Each basis function B_m is either a one-sided truncated function b as described by Equation (4.27), or a product of two or more of these functions

$$B_m(\mathbf{x}) = \prod_{j=1}^{J_m} \left[s_{jm} \cdot (x_{v(j,m)} - t_{jm}) \right]_+^q \quad (4.29)$$

where J_m is the number of factors in the m^{th} basis function, i.e. the number of functions b in the product. The parameter $s_{jm} = \pm 1$ and indicates the "left" or "right" version of the function, $x_{v(j,m)}$ denotes the v^{th} variable, where $1 \le v(j,m) \le k$ and k is the total number of

variables, and t_{jm} is the knot location for each of the corresponding variables. As previously, q indicates the power of the function.

Building a MARS metamodel is done in two steps. The first step starts with a_0, which is the mean of the response values in the fitting set. Basis functions B_m and B_{m+1} are then added in pairs to the model, choosing the ones that minimize a certain measure of lack of fit. Each new pair of basis functions consists of a term already in the model multiplied with the "left" and "right" version of a truncated power function b, respectively. The functions b are defined by a variable x_v and a knot location t. When adding a new pair of basis functions, the algorithm must therefore search over all combinations of the existing terms of the metamodel (to select the term to be used), all variables (to select the one for the new basis function), and all values of each variable (to find the knot location). For each of these combinations, the best set of coefficients a_m is found through a least square regression of the model response $\hat{\mathbf{y}}$ to the response from the fitting set $\mathbf{y}$. The process of adding terms to the model is continued until a pre-defined maximum number of terms are reached or until the improvement in lack of fit is sufficiently small. This so-called forward pass usually builds a model that overfits the data. The second step of the model building is therefore a backward pass where model terms are removed one by one, deleting the least effective term until the best metamodel is obtained. The lack of fit for the models is calculated using a modified form of generalized cross validation (see Section 4.4.3), which takes both the error and complexity of the model into account. More details can be found in Friedman (1991). The backward pass has the advantage that it can choose to delete any term except a_0. The forward pass can only add pairs of terms at each step, which are based on the terms already in the model.

A lot of searches have to be done during the model building. However, Jin et al. (2001) state that one of the advantages of the MARS metamodel, compared to Kriging, is the reduction in computational cost associated with building the model.

4.3.5 Support Vector Regression

Support vector regression (SVR) comes from the theory of support vector machines (SVM), which was developed at AT&T Bell Laboratories in the 1990s, see e.g Smola and Schölkopf (2004) for more details.

When it comes to metamodels, SVR can be seen to have similarities with other methods. The SVR metamodel can be described by the typical mathematical formulation

$$\hat{y}(\mathbf{x}) = b + \mathbf{w}^T\mathbf{Q}(\mathbf{x}) = b + \sum_{m=1}^{M} w_m Q_m(\mathbf{x}) \tag{4.30}$$

Hence, a sum of basis functions $\mathbf{Q} = [Q_1(\mathbf{x}), \ldots, Q_M(\mathbf{x})]^T$ with weights $\mathbf{w} = [w_1, \ldots, w_M]^T$

added to a base term b. The parameters b and w_m are to be estimated, but in a different way than the counterparts in other metamodels. The basis functions $\mathbf{Q}$ in the SVR model can be seen as a transformation of $\mathbf{x}$ into some feature space in which the model is linear, see Figure 4.12.

One of the main ideas with SVR is that a margin ε is given within which a difference between the fitting set responses and the metamodel prediction is accepted. This means that the fitting points that lie within the $\pm\varepsilon$ band (called the ε-tube) are ignored, and the metamodel is defined entirely by the points, called support vectors, that lie on or outside this region, see Figure 4.12. This can be useful when the fitting data has an element of random error due to numerical noise etc. A suitable value of ε might be found by a sensitivity study. In practical cases, however, the dataset is often not large enough to afford not to use all of the samples when building the metamodel. In addition, the time needed to train an SVR model is longer than what is required for many other metamodels.

Estimating the unknown parameters of an SVR metamodel is an optimization problem. The goal is to find a model that has at most a deviation of ε from the observed responses y_i ($i = 1, \ldots, n$) and at the same time minimizes the model complexity, i.e. makes the metamodel as flat as possible in feature space (Smola and Schölkopf, 2004). Flatness means that $\mathbf{w}$ should be small, which can be ensured by minimizing the vector norm $\|\mathbf{w}\|_2^2$. Since it might be impossible to find a solution that approximates all y_i with precision $\pm\varepsilon$ and that better predictions might be obtained if the possibility of outliers are allowed, slack variables ξ^+ and ξ^- can be introduced, see Figure 4.12. The optimization problem can then be stated as

$$\begin{aligned} \text{min} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) \\ \text{subject to} \quad & y_i - \mathbf{w}\cdot\mathbf{Q}(\mathbf{x}_i) - b \le \varepsilon + \xi_i^+ \\ & -y_i + \mathbf{w}\cdot\mathbf{Q}(\mathbf{x}_i) + b \le \varepsilon + \xi_i^- \\ & \xi_i^+, \xi_i^- \ge 0 \end{aligned} \tag{4.31}$$

This problem is a trade-off between model complexity and the degree to which errors larger than ε are tolerated. This trade-off is governed by the user defined constant $C > 0$, and this method of tolerating errors is known as the ε-insensitive loss function, see Figure 4.12. Other loss functions are also possible. The ε-insensitive loss function means that no loss will be associated to the points inside the ε-tube, while points outside will have a loss that increases linearly with a rate determined by C. A small constant will lead to a flatter prediction, i.e. more emphasis on minimizing $\|\mathbf{w}\|_2^2$, usually with fewer support vectors. A larger constant will lead to closer fitting of the data, i.e. more emphasis on minimizing $\sum(\xi^+ + \xi^-)$, usually with a larger number of support vectors. Although there might be an optimum value of C, the exact choice is not critical according to Forrester and Keane

(2009). It is therefore sufficient to try a few values of C of varying orders of magnitude and choose the one that provides the lowest error measure.

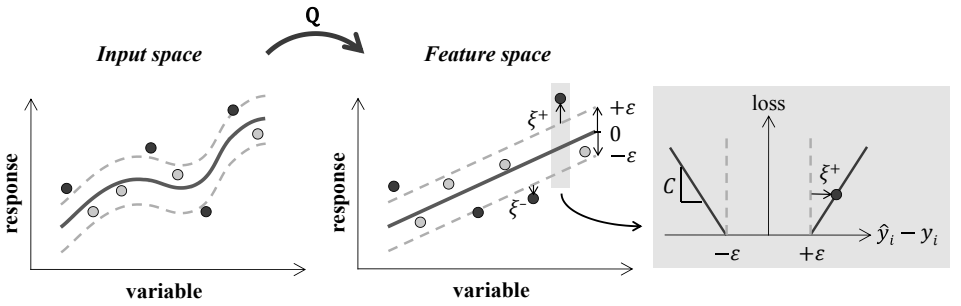

Figure 4.12 SVR metamodel in one design variable with support vectors marked with dark dots and the designs disregarded in the model build marked with light dots. The non-linear model is reduced to a linear model by the mapping **Q** from input space into feature space and the support vectors contribute to the cost by the ε-insensitive loss function.

In most cases, the optimization problem described by Equation (4.31) is more easily solved in its dual form, and it is therefore written as the minimization of the corresponding Lagrangian function L. At optimum, the partial derivatives of L with respect to its primal variables $\mathbf{w}$, b, ξ^+, and ξ^- must vanish, which leads to the optimization problem

$$\begin{aligned} \text{max} \quad & -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)k(\mathbf{x}_i,\mathbf{x}_j) - \varepsilon\sum_{i=1}^{n}(\alpha_i^+ + \alpha_i^-) + \sum_{i=1}^{n} y_i(\alpha_i^+ - \alpha_i^-) \\ \text{subject to} \quad & \sum_{i=1}^{n}(\alpha_i^+ - \alpha_i^-) = 0 \\ & (\alpha_i^+ - \alpha_i^-) \in [0, C] \end{aligned} \tag{4.32}$$

where α_i^+ and α_i^- are dual variables (Lagrange multipliers) and $k(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{Q}(\mathbf{x}_i) \cdot \mathbf{Q}(\mathbf{x}_j)$ represents the so-called kernel function. This problem can be solved using a quadratic programming algorithm to find the optimal choices of the dual variables. The kernel functions must have certain properties and possible choices include linear and Gaussian functions etc. as seen in Table 4.1 (Smola and Schölkopf, 2004).

The partial derivative of L with respect to $\mathbf{w}$ being zero yields $\mathbf{w} = \sum_{i=1}^{n}(\alpha_i^+ - \alpha_i^-)\mathbf{Q}(\mathbf{x}_i)$. Equation (4.30) can then be rewritten and provide the response in an unknown point $\mathbf{x}$ as

$$\hat{y}(\mathbf{x}) = b + \mathbf{w} \cdot \mathbf{Q}(\mathbf{x}) = b + \sum_{i=1}^{n}(\alpha_i^+ - \alpha_i^-)\mathbf{Q}(\mathbf{x}_i) \cdot \mathbf{Q}(\mathbf{x}) = b + \sum_{i=1}^{n}(\alpha_i^+ - \alpha_i^-)\, k(\mathbf{x}_i, \mathbf{x}) \tag{4.33}$$

The optimization problem in Equations (4.31) and (4.32) corresponds to finding the flattest function in feature space, not in input space. The base term is still unknown but could be determined from

$$b = y_j - \mathbf{w} \cdot \mathbf{Q}(\mathbf{x}_j) + \varepsilon = y_j - \sum_{i=1}^{n} (\alpha_i^+ - \alpha_i^-)\, k(\mathbf{x}_i, \mathbf{x}_j) + \varepsilon \;\; if\; 0 < \alpha_i^- < C \tag{4.34}$$

$$b = y_j - \mathbf{w} \cdot \mathbf{Q}(\mathbf{x}_j) - \varepsilon = y_j - \sum_{i=1}^{n} (\alpha_i^+ - \alpha_i^-)\, k(\mathbf{x}_i, \mathbf{x}_j) - \varepsilon \;\; if\; 0 < \alpha_i^+ < C \tag{4.35}$$

which means that b can be calculated for one or more $\alpha_i^{\pm}$ that fulfil the conditions. Better results are obtained for $\alpha_i^{\pm}$ not too close to the bounds according to Forrester and Keane (2009). The set of equations could also be solved via linear regression.

It can be seen that SVR methods produce RBF networks with all width parameters set to the same value and centres corresponding to the support vectors. The number of basis functions, i.e. hidden layer units, M in Equation (4.25), is the number of support vectors.

Table 4.1 Kernel functions for SVR where c, ϑ, and κ are constants.

Kernel function	**Mathematical description**
linear	$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
homogeneous polynomial of degree d	$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d$
inhomogeneous polynomial of degree d	$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d, c \geq 0$
Gaussian	$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-(\|\mathbf{x}_i - \mathbf{x}_j\|_2 / c)^2}$
Hyperbolic tangent	$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\vartheta + \kappa(\mathbf{x}_i \cdot \mathbf{x}_j))$

4.4 Metamodel Validation

The accuracy of a metamodel is influenced by the metamodel type as well as the quality and quantity of the dataset from which it is built. The quality of a metamodel cannot be described by only one single measure. Instead, there are several measures and methods that can be used for assessing the accuracy of a metamodel and comparing it to others.

4.4.1 Error Measures

One way to assess the accuracy of a metamodel is to study its residuals, i.e. the difference between the simulated value y_i and the predicted value from the metamodel $\hat{y}_i$. Small

residuals mean that the model reflects the dataset more accurately than if the residuals were larger, i.e. the fitting error is smaller. Several different error measures can be evaluated based on the residuals, e.g. the ***maximum absolute error*** (MAE), the ***average absolute error*** (AAE), the ***mean absolute percentage error*** (MAPE), the ***mean squared error*** (MSE), and ***the root mean squared error*** (RMSE).

$$MAE = max|y_i - \hat{y}_i|, \qquad i = 1, \dots, n \tag{4.36}$$

$$AAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \tag{4.37}$$

$$MAPE = \frac{\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i}}{n} \times 100\% \tag{4.38}$$

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{4.39}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{4.40}$$

where n is the number of samples in the fitting set. The smaller these error measures are, the smaller the fitting error is. The AAE, $MAPE$, MSE and $RMSE$ provide a measure of the overall accuracy, while the MAE is a measure of the local accuracy of the model. $RMSE$ is the most commonly used metric but can be biased as the residuals are not relatively measured. If the dataset contains both high and low response values, it might be desirable to study a relative error instead, i.e. an error measure that is independent of the magnitude of the response. The $MAPE$ measure takes this aspect into consideration.

Another commonly used statistic is the ***coefficient of determination*** R^2, which is a measure of how well the metamodel is able to capture the variability in the dataset.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.41}$$

where n is the number of design points and $\bar{y}$, $\hat{y}_i$, and y_i represent the mean, the predicted, and the actual response as defined in Figure 4.13. $R^2 \in [0\ 1]$, where 1.0 indicates a perfect fit. However, a high R^2 value can be deceiving if it is due to overfitting, which implies that the model will have poor prediction capabilities between the fitting points. Another occasion when the R^2 value can be misleading is when the response is insensitive to the studied variables, i.e. the metamodel equals the mean value of the observed responses. In this case R^2 will be close to 0 even for a well fitted model.

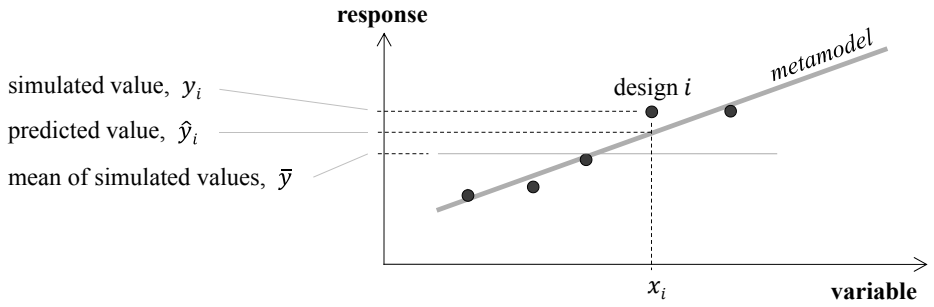

Figure 4.13 Description of definitions used for calculating error measures.

Some metamodels are interpolating the dataset, which means that there are no residuals and that R^2 equals 1.0. For the deterministic simulation case without random error or numerical noise, this is ideal. However, there is no guarantee that these interpolating metamodels are predicting the response between the known points better than other models. Furthermore, when numerical noise is present, it can be beneficial to filter the response by using an approximating model.

More interesting than studying the fitting error is often to estimate how well the metamodel can predict responses at unknown design points, i.e. study the prediction error. This can be done by evaluating the error measures mentioned above for a set of points that are not used to fit the model. It is essential that the validation set is large enough and spread over the design domain to provide a reliable picture of the accuracy. It is also important that the points in the validation set are not placed too close to the fitting points, since that can lead to an over-optimistic evaluation of the metamodel (Iooss et al., 2010).

4.4.2 Cross Validation

Another way of assessing the quality of a metamodel and comparing it to other models is called cross validation (CV). The methodology makes is possible to compare interpolating metamodels with approximating ones. In CV, the same dataset is used for fitting and validating the model. When the simulation time is long and the available data is limited, it can be desirable to use the complete dataset for fitting the metamodels rather than potentially lower the accuracy by leaving out a part of the set for validation.

In ***p-fold CV***, the dataset of n input-output data pairs is split into p different subsets. The metamodel is then fitted p times, each time leaving out one of the subsets. The omitted subset is used to evaluate the error measures of interest. A variation of the method is the ***leave-k-out CV***, in which all possible $\binom{n}{k}$ subsets of size k are left out, and the metamodel is fitted to the remaining set. Each time, the relevant error measures are evaluated at the

omitted points. This approach is computationally more expensive than the p-fold CV. However, for the special case when $k = 1$, called ***leave-one-out CV***, an estimation of the prediction error can be inexpensively computed for some metamodels, e.g. polynomial (Myers et al., 2008), Kriging (Martin and Simpson, 2004), and RBF models (Goel and Stander, 2009). The generalization error, i.e. the prediction error, for a leave-one-out calculation when the error is described by the MSE is represented by

$$MSE_{CV} = \frac{1}{n}\sum_{i=1}^{n} e_i^2 = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i^{(-i)}\right)^2 \tag{4.42}$$

where $\hat{y}_i^{(-i)}$ represents the prediction at $\mathbf{x}_i$ using the metamodel constructed utilizing all sample points except $(\mathbf{x}_i, y_i)$, e.g. see Forrester and Keane (2009).

The vector of leave-one-out errors needed to estimate the prediction error for a Kriging model fitted to all n points can be evaluated as

$$\mathbf{e} = \mathbf{Q}(\mathbf{R}^{-1}\mathbf{y} - \mathbf{R}^{-1}\mathbf{X}\mathbf{b}) \tag{4.43}$$

where $\mathbf{R}$ is the correlation matrix, $\mathbf{y}$ is the vector of observed responses, $\mathbf{b}$ is the vector of estimated regression coefficients, $\mathbf{X}$ is the model matrix, and $\mathbf{Q}$ is a diagonal matrix with elements that are the inverse of the diagonal elements of $\mathbf{R}^{-1}$.

For an RBF metamodel on the form $\hat{y}(\mathbf{x}) = b + \sum_{i=1}^{n_{RBF}} w_i f_i(\mathbf{x})$, the vector of leave-one-out errors can be evaluated as

$$\mathbf{e} = \left(diag(\mathbf{P})\right)^{-1}\mathbf{P}\mathbf{y} \tag{4.44}$$

where $\mathbf{y}$ is the vector of observed responses and $\mathbf{P}$ is the projection matrix, which is defined by

$$\mathbf{P} = \mathbf{I} - \mathbf{F}(\mathbf{F}^T\mathbf{F} + \mathbf{\Lambda})^{-1}\mathbf{F}^T \tag{4.45}$$

$\mathbf{F}$ is the design matrix constructed using the response of the radial functions at the design points such that $\mathbf{F}_{i1} = 1$, $\mathbf{F}_{ij+1} = f_j(\mathbf{x}_i)$, $i = 1, \ldots, n$ and $j = 1, \ldots, n_{RBF}$. $\mathbf{\Lambda}$ is a diagonal matrix, where Λ_{ii} , $i = 1, \ldots, n_{RBF}$, is the regularization parameter associated with the i^{th} weight as briefly mentioned at the end of Section 4.3.3.

It has been shown by Meckesheimer et al. (2002) that $k = 1$ in leave-k-out CV, i.e. leave-one-out, provides a good prediction error estimate for RBF and low-order polynomial metamodels. For Kriging models, the recommendation is instead to choose k as a function of the sample size, e.g. $k = 0.1n$ or $k = \sqrt{n}$.

The leave-one-out CV is a measure of how sensitive the metamodel is to lost information at its data points. An insensitive metamodel is not necessarily accurate and an accurate

model is not necessarily insensitive to lost information. Hence, the leave-one-out CV is not sufficient to measure metamodel accuracy, and validation with an additional dataset is therefore recommended (Lin, 2004). Small fitting sets, which are common in reality, are not suitable for CV (Stander et al., 2010). Data distribution could change considerably even when a small portion of the dataset is removed and used as a validation set. In addition, the CV approach is often expensive since it, in general, involves fitting of several metamodels for the same response. Nevertheless, CV could be the only practical way of obtaining information regarding the predictive capabilities of the metamodels in cases where the simulation budget is restricted and the detailed simulations are very cpu-expensive.

4.4.3 Generalized Cross Validation and Akaike's Final Prediction Error

Overfitting of a metamodel can lead to a model with a very small fitting error but a large prediction error. Overfitting generally occurs when a model is excessively complex, i.e. when it has too many parameters relative to the number of observations in the fitting set. Some measures of goodness of fit have therefore been developed that take both the residual error and the model complexity into account. One of these methods is the ***generalized cross validation*** (GCV) described by Craven and Wahba (1979), and another one is the ***final prediction error*** (FPE) defined by Akaike (1970). These error measures are evaluated for metamodels of different complexity fitted to the same dataset. The model with the lowest value should then be chosen as the one with the appropriate complexity. For a metamodel with a mean squared fitting error MSE, the corresponding GCV and FPE measures are defined by

$$MSE_{GCV} = \frac{MSE}{\left(1 - \frac{\nu}{n}\right)^2} \tag{4.46}$$

and

$$MSE_{FPE} = MSE\frac{1 + \frac{\nu}{n}}{1 - \frac{\nu}{n}} = MSE\frac{n + \nu}{n - \nu} \tag{4.47}$$

respectively (Stander et al., 2010). n is the number of fitting points, which should be large, and ν is the number of (effective) model parameters. In the original forms, valid for linear or unbiased models without regularization, ν is the number of model parameters. Otherwise, e.g. for neural network models, ν should be the number of effective model parameters, which could be estimated in different ways.

4.5 Optimization Algorithms

There are several different algorithms that can be used when solving a specific optimization problem. A local optimization algorithm only attempts to find a local optimum, and there is no guarantee that this optimum will also be the global one, unless very specific conditions are fulfilled. Thus, if the response has several local optima, different results can be obtained depending on the starting point. Most local optimization algorithms are gradient-based, i.e. they make use of gradient information to find the optimum solution, see e.g. Venter (2010). These techniques are popular because they are efficient, can solve problems with many design variables, and usually require little problem-specific parameter tuning. On the other hand, in addition to only finding local optima, they have difficulty solving discrete optimization problems and may be susceptible to numerical noise. When using a local optimization algorithm, a simple way of dealing with multiple local optima in the design space is to use a multi-start approach in which multiple local searches are performed from different starting points.

In most cases, the global optimum is requested, and what is often called a global optimization algorithm has a better chance of finding the global or near global optimum. Global optimization algorithms can be classified into two main categories: deterministic and stochastic (or heuristic) algorithms (Younis and Dong, 2010). Deterministic algorithms solve an optimization problem by generating a deterministic sequence of points converging to a globally optimal solution. Such algorithms behave predictable and given the same input, the algorithm will follow the same sequence of states and give the same result every time. The deterministic algorithms quickly converge to the global optimum but require the problem to have certain mathematical characteristics that it does not often possess.

The stochastic algorithms are based on a random generation of points that are used for non-linear local optimization search procedures. The algorithms are typically inspired by some phenomenon from nature, and have the advantage of being robust and well suited for discrete optimization problems. Compared to the deterministic algorithms, they usually have fewer restrictions on the mathematical characteristics of the problem, can search large design spaces, and do not require any gradient information. On the other hand, they cannot guarantee that an optimal solution is ever found and they often require many more objective function evaluations. Stochastic optimization algorithms are therefore particularly suitable for MBDO since the evaluations using metamodels take little time. Other drawbacks associated with stochastic algorithms include poor constraint-handling abilities, problem-specific parameter tuning, and limited problem size (Venter, 2010). Typical stochastic optimization algorithms include genetic algorithms, evolutionary strategies, particle swarm optimization, and simulated annealing.

Many of the stochastic optimization algorithms are population-based, i.e. a set of design points search for the optimum solution. These algorithms are particularly suitable for multi-objective optimization problems since a set of Pareto optimal points can be found in one single optimization run. One popular algorithm for solving MOO problems is the ***non-dominated sorting genetic algorithm*** (NSGA-II) developed by Deb et al. (2002).

Even if the stochastic optimization algorithms are associated with some drawbacks, they are often a suitable choice for the MDO process presented in Chapter 6. Some of these methods will therefore be presented in more detail in subsequent sections.

Since the different optimization algorithms have different benefits, hybrid optimization algorithms can be used in which the merits of different methods are taken advantage of. One example can be to initially use a stochastic optimization algorithm to find the vicinity of the global optimum, and then use a gradient-based algorithm to identify the optimum with greater accuracy.

4.5.1 Evolutionary Algorithms

Evolutionary algorithms (EAs) try to mimic biological evolution and are inspired by Darwin's principle of survival of the fittest. During the 1960s, different implementations of the basic idea were developed in different places. The algorithms are based on several iterations of the principal evolution cycle described in Figure 4.14 (Eiben and Smith, 2003). The process starts with a random population of candidate designs. The response value representing the objective function gives the fitness of each design in the population. Based on this fitness, some of the better candidates are chosen to seed the next generation. By applying recombination and mutation to these so-called parents, a set of new candidates, the offspring, is formed. The offspring then compete, based on their fitness and possibly age, with the parents for a place in the next generation. This process can be repeated until a candidate with sufficient fitness is found or until a previously defined computational limit is reached.

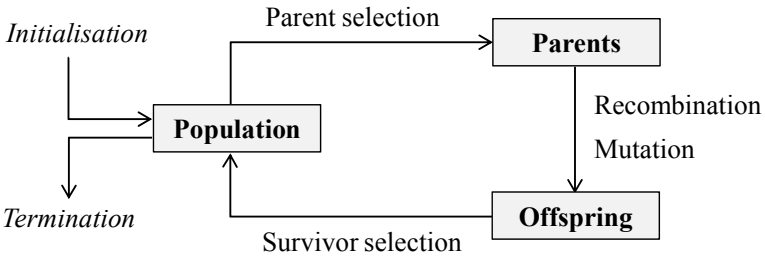

Figure 4.14 The basic evolution cycle followed by evolutionary algorithms.

Different variants of evolutionary algorithms follow the same basic cycle. They differ only in details related to a number of components, procedures and operators that must be specified in order to define a particular EA:

1. ***Representation***
   The candidate solutions are defined by a set of design variable settings and possibly additional information. These, so-called genes, must be represented in some way for the EA. This could for example be done by a string of binary code, a string of integers, or a string of real numbers.
2. ***Fitness function***
   The fitness function assigns a quality measure to the candidate solutions. This is normally the objective function or a simple transformation of it. If penalty functions are used to handle constraints the fitness is reduced for unfeasible solutions.
3. ***Population***
   A set of individuals, or candidate designs, forms a population. The number of individuals within the population, i.e. the population size, needs to be defined.
4. ***Parent selection mechanism***
   The role of parent selection is to distinguish among individuals based on their quality and to allow the better ones to become parents in the next generation. This selection is usually probabilistic so that high-quality individuals have higher chance of becoming parents than those with low quality. Nevertheless, low-quality individuals often still have a small chance of getting selected to avoid the algorithm from being trapped in a local optimum.
5. ***Variation operators***
   The role of variation operators are to create new individuals (offspring) from old ones (parents), i.e. generate new candidate designs. Recombination is applied to two or more selected candidates and results in one or more new candidates. Mutation is applied to one candidate and results in one new candidate. Both operators are stochastic and the outcome depends on a series of random choices. Several different versions exist for the various representations.
6. ***Survivor selection mechanism***
   The role of survivor selection is to select, based on their quality, the individuals that should form the next generation. Survivor selection is often deterministic, for instance ranking the individuals and selecting the top segment from parents and offspring (fitness biased) or selecting only from the offspring (age biased).

J. H. Holland (1992) is considered to be the pioneer of ***genetic algorithms*** (GAs), which are the most widely known type of evolutionary algorithms. There are several genetic algorithms that differ in representation, variation, and selection operators. What can be considered a classical GA has a binary representation, fitness proportionate parent

selection, a low probability of mutation, emphasis on genetically inspired recombination to generate new candidate solutions, and parents replaced by offspring in the next generation. This algorithm is commonly referred to as simple GA or canonical GA. Typically, recombination is done using 1-point crossover with a randomly selected crossover point, and mutation using bit flip, i.e. each bit is flipped (from 1 to 0 or 0 to 1) with a low probability, see Figure 4.15. It has been argued that real-coded GAs often produce better results than binary-coded GAs. However, it is problem dependant and can be more related to the selected crossover and mutation operators.

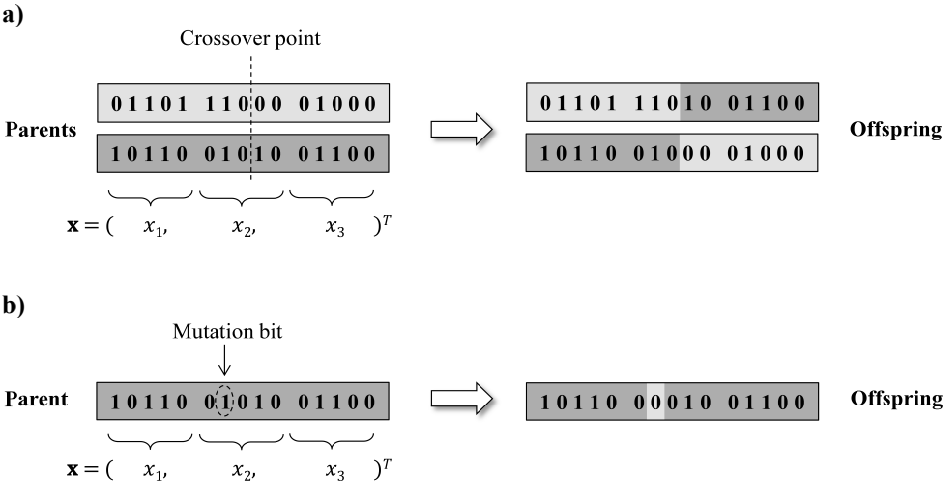

Figure 4.15 Typical variation operators used in simple GA for a three variable design in a binary string representation: **a)** recombination using 1-point crossover, and **b)** bit-flip mutation.

Evolution strategies (ESs) also belong to the EA family and were developed by I. Rechenberg and H.-P. Schwefel, see Beyer and Schwefel (2002). In the original ES algorithm, one parent individual is subjected to mutation to form one offspring and the better of these two individuals is chosen to form the next generation. Development of the method has now lead to more complex algorithms. General ESs have a real valued representation, random parent selection, and mutation as the primary operator for generating new candidate solutions. After creating λ offspring and calculating their fitness, the best μ are chosen deterministically, either from the offspring only, called (μ, λ) selection, or from the union of parents and offspring, called ($\mu + \lambda$) selection. Often (μ, λ) selection is preferred, especially if local optima exist. The value λ is generally much higher than the value μ, a ratio of 1 to 7 is recommended by Eiben and Smith (2003). Most ESs are self-adaptive, which means that some parameters are included in the represen-

tation of the individuals and co-evolve with the solutions so that the algorithm performs better. Recombination is commonly discrete for the variable part and intermediary for the strategy parameter part, see Figure 4.16. Usually, global recombination is used, i.e. the parents are drawn randomly from the population for each position *i* so that more than two individuals commonly contribute to the offspring. Mutation is often done by Gaussian perturbation, which means that each variable is changed a small amount randomly drawn from a normal distribution.

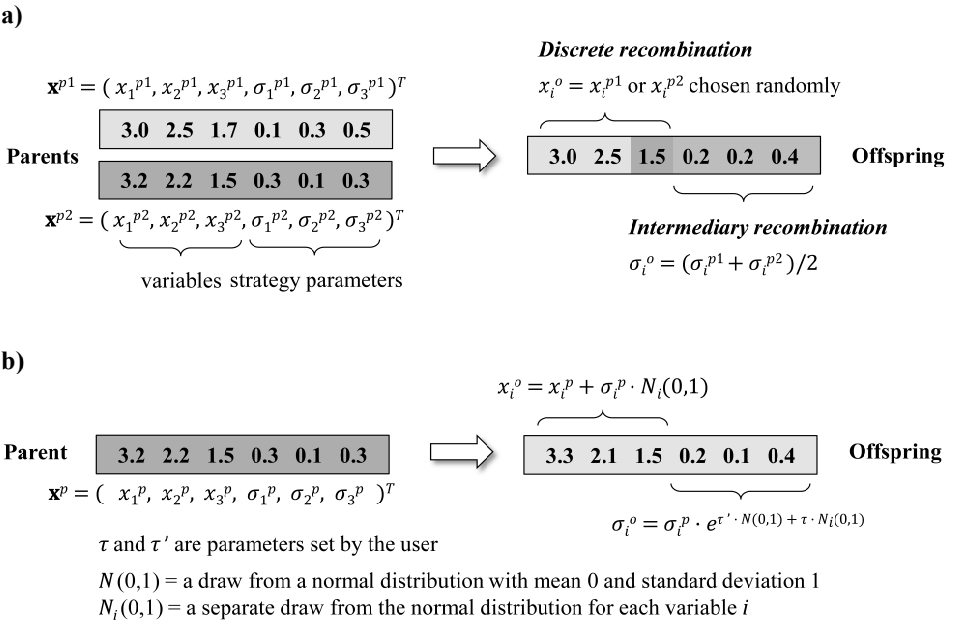

Figure 4.16 Typical variation operators used in an ES for a three variable design in a real valued representation: **a)** discrete and intermediary recombination, and **b)** mutation by Gaussian perturbation.

A comparison between GAs and ESs is presented in Table 4.2. In general, GAs are considered more likely to find the global optimum, while ESs are considered faster. A general recommendation is therefore to use a GA if it is important to find the global optimum, while an ES should be used if speed is important and a "good enough" solution is acceptable. However, the results depend on the algorithm settings and the problem at hand, and there is no generally accepted conclusion on the superiority of any of the algorithms. Instead, the merits of both algorithms could be taken advantage of if they are used together (Hwang and Jang, 2008).

Table 4.2 Overview of typical features of genetic algorithms and evolution strategies according to Eiben and Smith (2003).

	Genetic algorithms	**Evolution strategies**
Typical representation	Strings of a finite alphabet	Strings of real numbers
Role of recombination	Primary variation operator	Secondary variation operator
Role of mutation	Secondary variation operator	Primary and sometimes the only variation operator
Parent selection	Random, biased by fitness	Random, uniform
Survivor selection	All individuals replaced or deterministic, biased by fitness	Deterministic, biased by fitness

Constraints are often enforced by using penalty functions that reduce the fitness of unfeasible solutions. Preferably, the fitness is reduced in proportion to the number of constraints that are violated. A good idea is often also to reduce the fitness in proportion to the distance from the feasible region. The penalty functions are sometimes set so large that unfeasible solutions will not survive. Occasionally the penalty functions are allowed to change over time and even adapt to the progress of the algorithm. There are also other techniques to handle constraints. One of them is to use a repair function that modifies an unfeasible solution into a feasible one.

4.5.2 Particle Swarm Optimization

Swarm algorithms are based on the idea of swarm intelligence, i.e. the collective intelligence that emerges from a group of individuals, and are inspired by the behaviour of organisms that live and interact in nature within large groups. One of the most well-known algorithms is ***particle swarm optimization*** (PSO), which imitates for example a flock of birds. Hence, swarm algorithms are population-based algorithms like the evolutionary algorithms.

Particle swarm optimization was introduced by J. Kennedy and R. Eberhart after studying the social behaviour of birds (Kennedy and Eberhart, 1995). To search for food, each member of a flock of birds determines its velocity based on their personal experience as well as information gained through interactions with other members of the flock. The same ideas apply to PSO, in which the population, called swarm, converges to the optimum using information gained from each individual, referred to as particle, and from the information gained by the swarm as a whole. A basic PSO algorithm has a very simple formulation that is easy to implement and modify. The algorithm starts by initializing a swarm of particles with randomly chosen velocity and position within the design space. The position of each particle is then updated from one iteration to the next using the simple formula

$$\mathbf{x}_i^{q+1} = \mathbf{x}_i^q + \mathbf{v}_i^q \Delta t \tag{4.48}$$

where i refers to the i^{th} particle in the swarm, q to the q^{th} iteration and $\mathbf{v}_i{}^q$ to the velocity. The time increment Δt is usually set to one and the velocity vector is updated in each iteration using

$$\mathbf{v}_i^{q+1} = w\mathbf{v}_i^q + c_1 r_1 \frac{(\mathbf{p}_i - \mathbf{x}_i^q)}{\Delta t} + c_2 r_2 \frac{(\mathbf{p}_g - \mathbf{x}_i^q)}{\Delta t} \tag{4.49}$$

where w is the inertia parameter, r_1 and r_2 are random numbers between 0 and 1, c_1 and c_2 are the trust parameters, $\mathbf{p}_i$ is the best point found so far by the i^{th} particle, and $\mathbf{p}_g$ is the best point found by the swarm. Thus, the user needs to select or tune the values of w, c_1 and c_2, and decide on the number of particles in the swarm, as well as how many iterations that should be performed. The inertia parameter w controls the search behaviour of the algorithm. Larger values (around 1.4) result in a more global search, while smaller values (around 0.5) result in a more local search (Venter, 2010). The c_1 trust parameter indicates how much the particle trusts itself, while c_2 specifies how much the particle trusts the group. Recommended values are $c_1 = c_2 = 2$ (Venter, 2010). Finally, $\mathbf{p}_g$ can be selected to represent either the best point in a small subset of particles or the best point in the whole swarm.

The original PSO algorithm has been developed and enhanced, and different versions have been applied to different types of optimization problems. Constraints can be handled by penalty methods, as described in the previous section. Another simple approach is to use strategies that preserve feasibility. Hu et al. (2003) describe a method where each particle is initialized repeatedly until it satisfies all the constraints and where the particles then search the whole space but only keep the feasible solutions in their memory.

4.5.3 Simulated Annealing

Simulated annealing (SA) is a global stochastic optimization algorithm that mimics the metallurgical annealing process, i.e. heating and controlled cooling of a metal to increase the size of its crystals and reduce their defects. The algorithm was developed by S. Kirkpatrick and co-workers, and exploits the analogy with a metal that cools and freezes into a minimum energy crystalline structure (Kirkpatrick et al., 1983). In SA, the objective function of the optimization problem is seen as the internal energy of the metal during annealing. The idea is to start at a high temperature, which is slowly reduced so that the system goes through different energy states in the search of the lowest state representing the global minimum of the optimization problem. When annealing metals, the initial temperature must not be too low and the cooling must be done sufficiently slowly to avoid the system from getting stuck in a meta-stable non-crystalline state representing a local minimum of energy. The same principles apply to simulated annealing in the process of

finding the solution to an optimization problem.

The strength of the SA algorithm is its ability to deal with highly non-linear, chaotic, and noisy objective functions, as well as a large number of constraints. On the other hand, a major drawback is the lack of clear trade-off between the quality of a solution and the time required to locate the solution, which leads to long computation times (Younis and Dong, 2010). Different modifications to the original algorithm have been proposed to improve the speed of convergence. One of these is the ***very fast simulated re-annealing*** algorithm, also known as ***adaptive simulated annealing*** (ASA) (Ingber, 1996).

Simulated annealing algorithms can, in general, be described by the following steps (Stander et al., 2010):

1. ***Initialisation***
   The search starts at iteration $q = 0$ by identifying a starting design, also called starting state, $\mathbf{x}^{(0)}$ from the set of all possible designs $\mathbf{X}$ and calculating the corresponding energy $E^{(0)} = E(\mathbf{x})$. The set of checked points $\mathbf{X}^{(0)} = \{\mathbf{x}^{(0)}\}$ now includes only the starting design. The temperature is initialized at a high value $T^{(0)} = T^{max}$ and a cooling schedule C, an acceptance function A, and a stopping criterion are defined.

2. ***Sampling***
   A new sampling point $\mathbf{x}' \in \mathbf{X}$ is selected using a sampling distribution $D(\mathbf{X}^{(q)})$, and the corresponding energy $E' = E(\mathbf{x}')$ is calculated. The set of checked points $\mathbf{X}^{(q+1)} = \mathbf{X}^{(q)} \cup \{\mathbf{x}'\}$ now contains $q + 2$ designs.

3. ***Acceptance check***
   A random number ζ is sampled from the uniform distribution [0 1] and

$$\mathbf{x}^{(q+1)} = \begin{cases} \mathbf{x}' & if\ \zeta \leq A(E', E^{(q)}, T^{(q)}) \\ \mathbf{x}^{q} & otherwise \end{cases} \tag{4.50}$$

   where A is the acceptance function that determines if the new point is accepted. The most commonly used acceptance function is the Metropolis criterion

$$A\left(E', E^{(q)}, T^{(q)}\right) = min\left\{1,\ e^{\frac{-(E'-E^{(q)})}{T^{(q)}}}\right\} \tag{4.51}$$

4. ***Temperature update***
   The cooling schedule $T^{(q+1)} = C\ (\mathbf{X}^{(q+1)},\ T^{(q)})$ is applied to the temperature. It has been proven that a global minimum will be obtained if the cooling is made sufficiently slowly (Geman and Geman, 1984).

5. ***Convergence check***
   The search is ended if the stopping criterion is met, otherwise $q = q + 1$ and the search continues at step 2. Usually, the search is stopped when there is no noticeable improvement over a number of iterations, when the number of iterations has reached a predefined value, or when the temperature has fallen to a desired level.

It is obvious that the efficiency of the algorithm depends on the appropriate choices of the mechanisms to generate new candidate states D, the cooling schedule C, the acceptance criterion A, and the stopping criterion. The choices of D and C are generally the most important issues in defining an SA algorithm and they are strongly interrelated. The next candidate design $\mathbf{x}'$ is usually selected randomly in the neighbourhood of the current design $\mathbf{x}$, with the same probability for all neighbours. The size of the neighbourhood is typically selected based on the idea that the algorithm should have more freedom when the current energy is far from the global optimum. Larger step sizes are therefore allowed initially. However, a more complicated, non-uniform selection procedure is used in adaptive simulated annealing to allow much faster cooling rates (Stander et al., 2010). The basic idea of the cooling schedule is to start at a high temperature and then gradually drop the temperature to zero. The primary goal is to quickly reach a temperature with low energies, but where it is still possible to explore different areas of the design space. Thereafter, the SA algorithm lowers the temperature slowly until the system freezes and no further changes occur.

Simulated annealing algorithms generally handle constraints by penalty methods similar to the ones described in Section 4.5.1, i.e. the energy for unfeasible solutions is increased so that the probability of selecting such designs is reduced.

Hill-climbing is a very simple local optimization algorithm where new candidate designs are iteratively tested in the region of the current design and adopted if they are better. This enables the algorithm to climb uphill until a local maximum is found. A similar technique could be used to find a local minimum. Simulated annealing differs from these simple algorithms in that new candidate solutions can be chosen at a certain probability even if they are worse than the previous one, i.e. have higher energy. A new worse solution is more likely to be chosen early in the search when the temperature is high and if the difference in energy is small. Simulated annealing hence goes from being similar to a random search initially, with the aim of finding the region of the global optimum, to being very similar to "Hill-climbing" in order to locate the minimum more exactly.

5 Multidisciplinary Design Optimization

Historically, the roots of multidisciplinary design optimization can be found in structural optimization, mainly within the aerospace industry (Agte et al., 2010). Disciplines strongly interacting with structural parts were first included in the optimization problem, making it multidisciplinary. The development has then been heading towards incorporating whole systems in the MDO studies, i.e. also including design variables important for other disciplines than the structural ones.

The development of MDO can be described in terms of three generations (Kroo and Manning, 2000), see Figure 5.1. Initially, when the problem size was limited, all disciplinary analyses were integrated directly with an optimizer. To be able to perform more extensive MDO studies, analyses were distributed in the second generation of MDO methods. The first two generations are so-called single-level optimization methods, i.e. they rely on a central optimizer to coordinate the optimization and make all design decisions. When MDO was applied to even larger problems involving several departments within a company, it was found unpractical to rely on a central optimizer as the only decision-maker. The third generation of MDO methods therefore consists of the so-called multi-level optimization methods, where the optimization process as such is distributed. The distribution of decisions more closely resembles the standard product development process where different groups are responsible for the development of different parts and aspects of the product.

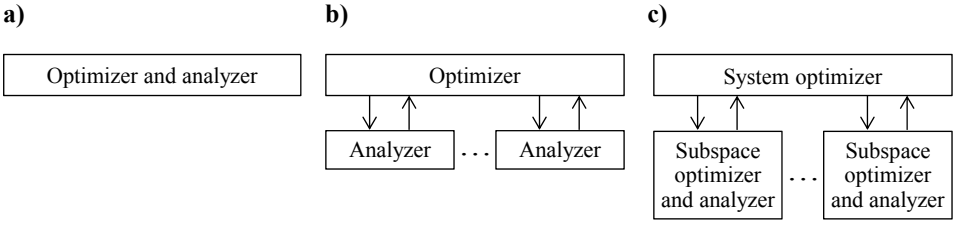

Figure 5.1 The three generations of MDO methods: **a)** single-level optimization with integrated analyses, **b)** single-level optimization with distributed analyses, and **c)** multi-level optimization.

As the first generation of MDO methods are unsuitable for large-scale applications, single-level methods will refer to the second generation of MDO methods in the following.

5.1 Single-Level Methods

Common for single-level optimization methods is a central optimizer that makes all design decisions. The most common and basic single-level approach is the ***multidisciplinary feasible*** (MDF) formulation (Cramer et al., 1994). In the MDF formulation, the optimizer requests the values for the objective and constraint functions for different sets of design variables from a so-called system analyzer. The system analyzer enforces multidisciplinary consistency from the subspace analyzers, i.e. finds a consistent set of coupling variables, for each set of design variables. This is usually done iteratively using either fixed-point iteration or Newton's method (Balling and Sobieszczanski-Sobieski, 1994). There are a number of drawbacks related to efficiency and robustness associated with the MDF formulation (Allison et al., 2005). For example, the parallelism is limited when the system analyzer tries to achieve multidisciplinary consistency, and the optimizer may fail to find the optimal design if the system analyzer has convergence problems.

The ***individual discipline feasible*** (IDF) formulation is an alternative single-level approach proposed by Cramer et al. (1994) where multidisciplinary consistency is only enforced at optimum. Here, the optimizer sends the design variables and estimations of the coupling variables directly to the subspace analyzers. The subspace analyzers return updated coupling variables as well as contributions to the global objective and constraint functions to the optimizer. An additional constraint is introduced for each coupling variable to drive the optimization process towards multidisciplinary consistency at optimum. In this approach, the subspace analyzers are thus decoupled and the iterative process needed to find multidisciplinary consistent designs at every call from the optimizer is avoided. However, since an additional variable and an extra constraint are introduced for each coupling variable, the method is most efficient for problems with few coupling variables.

The MDF and the IDF methods differ in the handling of coupling variables, i.e. how multidisciplinary consistency is enforced. However, the two formulations coincide for MDO problems lacking coupling variables.

5.2 Multi-Level Methods

In multi-level optimization methods, the decision-making process is distributed and a system optimizer communicates with a number of subspace optimizers. Several multi-level optimization formulations can be found in the literature and some of the most well-

known ones are briefly presented here. More detailed descriptions can be found in e.g. Ryberg et al. (2012).

Concurrent subspace optimization (CSSO) was first introduced by J. Sobieszczanski-Sobieski (1988). It is an iterative approach that starts with finding a multidisciplinary consistent solution for the initial design. The main idea is then to distribute each shared variable to the subspace whose objective and constraint functions it affects the most. Next, all subspaces are optimized with respect to its local variables and a subset of its shared variables, while all other variables are held constant. The formulation of each subspace optimization problem includes minimization of the objective function subject to one combined local constraint and to approximations of the combined constraints from the other subspaces. The responsibility for fulfilling the constraints is thus shared between all the subspace optimizers, and the distribution of the responsibility is governed by a system coordinator. The new design point is simply the combination of optimized variables from the different subspaces, and this point is not necessarily feasible (Pan and Diaz, 1989). Finally, the system coordinator redistributes the responsibility for the different constraints for the next iteration and the process is continued until convergence is reached. Unfortunately, the method can experience convergence issues and may fail to solve some simple problems (Shankar et al., 1993).

A more recent version of CSSO was developed by Renaud and Gabriele (1991, 1993, 1994) in a series of articles, and many subsequent approaches are based on their work. In their formulation, the subspace optimizations are followed by the solution of an approximation of the global optimization problem. This approximation is constructed around the combination of optimized variables from the different subspaces. The obtained optimum is then the design vector input to the next iteration. All variables are consequently dealt with at the system level. This restricts the autonomy of the groups responsible for each subspace, which is the main motivation for using a multi-level method.

Bilevel integrated system synthesis (BLISS) was first presented by J. Sobieszczanski-Sobieski et al. (1998) and the method has some similarities with CSSO. The original implementation concerns four coupled subspaces of a supersonic business jet: structures, aerodynamics, propulsion, and aircraft range. The method is iterative and optimizes the design in two main steps. First, subspace optimizations are performed in parallel with respect to the local variables subject to local constraints. Next, the system optimizer finds the best design with respect to the shared variables. A linear approximation of the global objective function is constructed and split into objectives for the system and subspace optimization problems. Normally, the system optimization problem is considered to be an unconstrained problem. However, if the constraints in the subspace optimizations depend more strongly on the shared and coupling variables than on the local variables, they might need to be included in the system optimization, turning the system optimization problem into a constrained one. The BLISS procedure separates the optimization with respect to

the local and shared variables, and sometimes a different solution can be obtained compared to the case when all variables are optimized simultaneously (Kodiyalam and Sobieszczanski-Sobieski, 2000).

A reformulation of the original method, referred to in the literature as ***BLISS 2000***, or simply BLISS, was presented by Sobieszczanski-Sobieski et al. (2003). The key concept in BLISS 2000 is the use of surrogate models to represent optimized subspaces. To create these surrogate models, a DOE is created and a number of subspace optimization problems are solved with respect to the local variables. In each subspace optimization problem, the sum of the coupling variables output from that specific subspace multiplied with weighting coefficients is minimized subject to local constraints. The resulting surrogate models represent the coupling variables output from each subspace as functions of the shared variables, the coupling variables input to that subspace, and the weighting coefficients. Polynomial surrogate models are used in the original version of BLISS 2000, but each subspace could, in principle, be given the freedom to choose their own surrogate model. The system optimizer uses the surrogate models to minimize the global objective subject to consistency constraints. The BLISS 2000 formulation was developed to handle coupled subspaces and is not relevant for problems lacking coupling variables since the subspace objective functions then no longer exist.

An early description of ***Collaborative optimization*** (CO) was published by Kroo et al. (1994), and it was further refined by Braun (1996). Collaborative optimization can handle coupling variables and has mainly been used for aerospace applications. In CO, the system optimizer is in charge of target values of the shared and coupling variables. The subspaces are given local copies of these variables, which they have the freedom to change during the optimization process. The local copies converge towards the target values at optimum, i.e. a consistent design is obtained. The system optimizer minimizes the global objective function subject to constraints that ensure a consistent design. The subspace optimizers minimize the deviation from consistency subject to local constraints. There are a number of numerical problems associated with CO when used in combination with gradient-based algorithms (DeMiguel and Murray, 2000; Alexandrov and Lewis, 2002). These problems hinder convergence proofs and have an unfavourable effect on the convergence rate.

A number of attempts to modify the CO formulation in order to overcome the numerical difficulties are documented in the literature. These include the introduction of polynomial surrogate models to represent the subspace objective functions (Sobieski and Kroo, 2000), modified collaborative optimization (DeMiguel and Murray, 2000), and enhanced collaborative optimization (Rooth, 2008). In enhanced collaborative optimization (ECO), the goal of the system optimizer is to find a consistent design. There are no constraints at the system level, which makes the system optimization problem easy to solve. The objective functions of the subspaces contain the global objective in addition to measures of the deviation from consistency. It is intuitively more appealing for the subspaces to work

towards minimizing a global objective, rather than towards minimizing a deviation from consistency as is done in the original CO formulation. The subspaces are subject to local constraints as well as to linearized versions of the constraints in the other subspaces. The inclusion of the latter constraints provides a direct understanding of the preferences of the other subspaces, unlike CO, where this knowledge is only obtained indirectly from the system optimizer. However, the complexity of ECO is a major drawback.

Analytical target cascading (ATC) was developed for automotive applications (Kim, 2001). It was originally intended as a product development tool for propagating targets, i.e. convert targets on the overall system to targets on smaller parts of the system. However, it can also be used for optimization if the targets are unattainable. Analytical target cascading was established for an arbitrary number of levels, but a formulation with two levels, as in the previously presented methods, is also possible. The objective functions of the optimization problems on all levels consist of terms that minimize the unattainability of the target and terms that ensure consistency (Michalek and Papalambros, 2005b). In each of these problems, there are local variables and local constraints. Shared and coupling variables are handled in a fashion similar to CO, i.e. an upper level optimizer has target values and the lower level optimizers have local copies of these variables. Normally, consistency is obtained by including the square of the L_2-norm of the deviation from consistency multiplied by penalty weights in the objective functions. In this formulation, it is important to choose the penalty weights appropriately (Michalek and Papalambros, 2005a). Too small weights can yield solutions far from the solution of the original problem, and too large weights can cause numerical problems. Other types of penalty functions have therefore been proposed in the literature, see e.g. Tosserams et al. (2006).

5.3 Suitable Methods for Automotive Structures

To be able to implement MDO into an automotive product development process and use it for large-scale applications, the groups involved need to work concurrently and autonomously. This is not different compared to the situation in the aerospace industry, and the development of multi-level MDO methods have therefore aimed at distributing the design process. However, most multi-level methods were initially developed for direct optimization of coupled disciplines, and the gain in concurrency and autonomy was obtained at the cost of added complexity. These methods can be complicated both to implement and to use. A multi-level MDO problem can also become less transparent than the corresponding single-level problem. One example is when the local objective functions do not mirror the global objective function, which makes it difficult for the individual groups to grasp the global goal of the optimization process. Some of the methods are also associated with numerical problems when used in combination with gradient-based optimization algo-

rithms. Furthermore, it is not unusual that multi-level MDO methods require more computational resources than single-level methods.

When performing optimization of automotive structures, it is often considered necessary to use metamodels to reduce the required computational effort. It is possible to use metamodels in both single-level and multi-level MDO methods. However, the effects are more interesting and important when metamodels are used in combination with single-level methods. In both cases, the effect from computational efficiency is obtained, but in addition, the drawback of limited autonomy and concurrency associated with single-level methods can be relieved by the introduction of metamodels. In metamodel-based design optimization, the main computational effort is spent on building the metamodels. During this process, the groups involved can work concurrently and autonomously using their preferred methods and tools. The issue of not participating in design decisions when using single-level methods can also partly be compensated by involving the different groups in the setup of the optimization problem and in the assessment of the results. However, since the optimization is done on a system level, the individual groups cannot govern the choice of optimization methods and tools. In addition, groups that have inexpensive simulations either have to create metamodels and introduce an unnecessary source of error, or let the central optimizer call their analyzers directly and give up more of their autonomy. Despite these drawbacks, a single-level method in combination with the use of metamodels is often the most convenient way to solve MDO problems for automotive structural applications that involve computationally expensive simulations. The complexity of the multi-level methods and lack of readily available software make them a less attractive alternative.

6 An MDO Process for Automotive Structures

Multidisciplinary design optimization is not yet a standard methodology for automotive structures, but there are obvious benefits if this could be achieved. In order to implement MDO into the product development process, there are a number of aspects that need to be considered, both related to the characteristics of the problems and the implementation of the MDO method itself. An MBDO approach based on global metamodels turns out to be an appropriate methodology for MDO of automotive structures. Its suitability and efficiency for general automotive structural problems can clearly be demonstrated in a test problem, which resembles a typical MDO problem from the automotive industry.

6.1 Requirements

To solve a large-scale automotive MDO problem generally involves several groups within a company. A process that fits the company organization and product development process is therefore needed for MDO to be a part of the daily work. It is also important to make use of the existing expertise within the company, i.e. let the experts take part in the design decisions and let them use their own methods and tools. In addition, MDO studies must be realized in a reasonable time in order to fit the product development process. Altogether, this can be summarized in

> ***Requirement 1:*** *The different groups need to work concurrently and autonomously.*

The MDO process also needs to be computationally efficient. It is important to limit the time needed for the study and not exceed the available computer resources. Multidisciplinary design optimization requires evaluations of many different variable settings for all the included loadcases, and the detailed simulation models are commonly computationally expensive to evaluate. Consequently, is it often efficient to use metamodels in the optimization process in order to reduce the required computational resources. Hence,

> ***Requirement 2:*** *The process must be able to incorporate metamodels.*

To solve an MDO problem that involves coupling variables is significantly more complicated than if only shared variables are considered. Since coupling variables are rare when solving MDO problems for automotive structures they can be neglected in the proposed process. Consequently,

Requirement 3: *Coupling variables need not to be considered.*

The nature of MDO studies can differ substantially, e.g. be extensive or limited depending on the number of variables and loadcases, and have one or several objectives. Variations are always present in reality, and there is an interest to consider uncertainties in the design also during optimization. It is important that the process can handle as many of these variants as possible. Thus,

Requirement 4: *The process must be flexible and able to consider multiple objectives and robustness.*

The main driver for implementing a multi-level optimization method is to distribute the design process (Kroo and Manning, 2000). However, multi-level methods complicate the solution process and to justify their use, the advantages must be greater than the cost. It was concluded in Section 5.3 that a single-level method in combination with the use of metamodels is the most straightforward way of solving automotive MDO problems without coupling variables. The drawback of not making design decisions can be relieved by involving the different groups in the setup of the optimization problem and in the assessment of the results. The benefits of a multi-level method are thus not considered to compensate for the drawbacks. Hence,

Requirement 5: *A single-level MDO method should be used.*

6.2 Process description

A metamodel-based design optimization approach based on global metamodels, which aim at capturing the responses over the complete design space, fulfils the requirements outlined in the previous section. The process is illustrated in Figure 6.1 and can be used for both optimization of characteristics from one discipline and for multidisciplinary design optimization. The process has six distinctive steps, which are described in more detail below. However, the solution of an optimization problem is not only an execution of these steps in sequence. If the results from one step are found to be unsatisfactory, it is often necessary to go back and improve one of the previous steps, resulting in an iterative workflow.

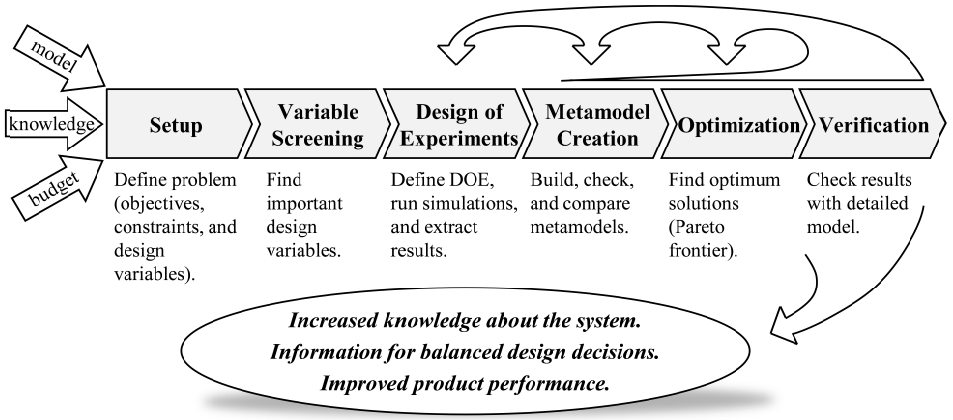

Figure 6.1 A metamodel-based multidisciplinary design optimization process.

The aim of the described optimization process is to increase the knowledge of the system so that balanced design decisions can be made, which result in improved product performance. However, there are a number of prerequisites for the MDO study to be successful. Firstly, stable simulation models that accurately capture the behaviour of the product must be available. A stable model can be run without failures when the variables are changed within the design space. Additionally, the variations in output are mainly caused by changes in input and are not due to numerical noise. It is also essential to have a thorough understanding of the simulation models in order to know their limitations and be able to assess the obtained results. Moreover, it is important to be aware of the simulation budget to make a decision regarding the amount of simulations that can be afforded in the different steps of the process. The six steps of the process are:

Step 1: Setup

First, the scope of the problem must be specified, i.e. the appropriate loadcases should be selected and the problem should be mathematically defined. The mathematical definition includes selection of the objective or objectives; the design variables and the range within which they are allowed to vary, i.e. the design space; and the performance constraints. The objectives and performance constraints are responses from the simulation models or derived from the responses.

Step 2: Variable screening

The number of simulations needed to build accurate metamodels is highly dependent on the number of design variables. The next logical step is therefore to identify the variables that influence the studied responses the most, and only consider these in the subsequent steps.

This can be done using a limited number of detailed simulations and one of several available screening methods.

Step 3: Design of experiments

Next, a suitable DOE needs to be selected and the simulations with the detailed models must be run. Afterwards, the relevant responses can be extracted resulting in a database of input (variable settings) and output (response values). During the screening, different sets of variables are identified as important for different loadcases. For each loadcase, only the important variables are included in the DOE, and all the other variables are held constant.

Step 4: Metamodel creation

The metamodels can now be built from the available dataset and their accuracy should be carefully checked. It is often useful to build more than one metamodel for each response. Since it can be hard to compare the accuracy between different metamodel types and identify the most accurate metamodels, it can also be beneficial to use different sets of metamodels in the next step.

Step 5: Optimization

When the metamodels are found to be satisfactory, the optimization can be performed. In general, optimization algorithms cannot guarantee that the global optimum is obtained. It is therefore preferable to use more than one optimization algorithms for each set of metamodels. Several design proposals are then obtained and can be studied further.

Step 6: Verification

Based on the optimization results, one or several potential designs can be selected and verified using the detailed simulation models. Differences between results from the optimization study and results from the verification simulations are caused by inaccurate metamodels or improper selection of variables during the screening. Sometimes, there are large constraint margins or no feasible design is found. Then, manual adjustments of the design proposals based on information from the metamodels and the screening can improve the results. However, if the discrepancies are large, it might be necessary to go back and improve the step causing the issues.

6.3 Application Example

The described process can easily be demonstrated in an example. Here, an MDO problem is used that mimics a common weight optimization study of an automotive body structure similar to the typical automotive MDO example found in the literature. Four loadcases are studied, three that resemble crashworthiness configurations and one NVH-related assessment. The goal is to minimize the mass subject to two performance constraints from each loadcase, which ensure that the performance is maintained at least on the same level as in the nominal model. There are originally 25 variables representing different sheet thicknesses in the structure and these add up to a variable mass of 8.38 kg. The studied structure and an illustration of the different loadcases are found in Figure 6.2.

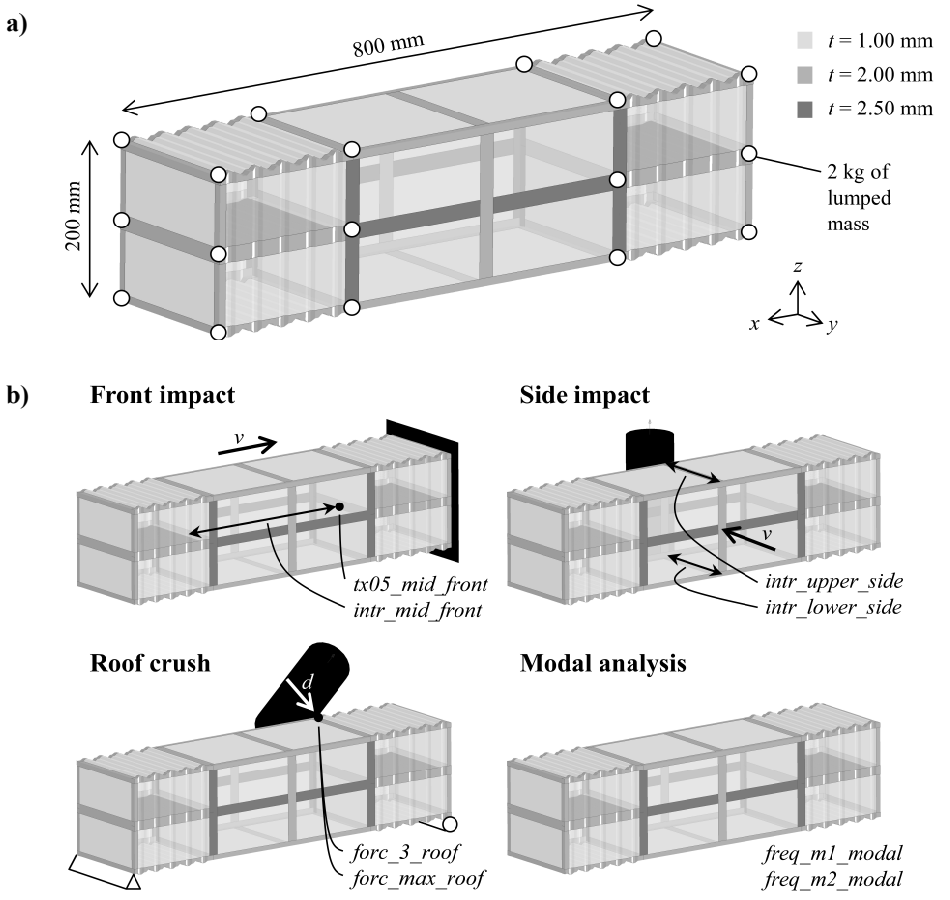

Figure 6.2 a) Geometry of the studied structure. **b)** Loadcases with the measured responses indicated.

When the described process is followed, there are many choices that must be made, e.g. related to software, screening methods, DOEs, metamodels, and optimization algorithms. The example presented here is only an illustration of the proposed process and the selection of methods for other studies should be related to the nature of the considered problem, rather than a copy of the methods used here. The execution and results for the different steps are briefly described below, and a more detailed description can be found in Paper II.

The screening is performed using a simple technique in which one variable at a time is set to the lowest value, while the other variables are left at their nominal values. It is then possible to identify the variables that influence the different responses by comparing the obtained results with the results from the nominal design. Here, the selection is done based on a global sensitivity analysis (see Section 4.2) and a criterion that the difference in result compared to the nominal design must not be too large. In this way, the number of variables is reduced from the original 25 to 15, 7, 11, and 12, respectively, for the four different loadcases. It is noted that three of the variables are not considered to be important for any loadcase and these variables are therefore set to their minimum values for the rest of the study.

Among the different space-filling algorithms available, a maximin design is chosen, (see Section 4.1). The sample size is initially set to $3n_i$, where i represents the different loadcases and n_i is the number of variables for loadcase i. Based on the estimated accuracy of the metamodels, it is judged whether or not a larger DOE sample size is needed for the different loadcases. Additional design points are then added sequentially in groups of n_i, up to a maximum of $6n_i$ design points for loadcase i. The final number of simulations for the different loadcases is presented in Table 6.1. The variables not included in the different DOEs are seen as constants and set to their nominal values.

Table 6.1 Final DOE sizes for the different loadcases.

Loadcase	Number of variables, n_i	DOE size	Number of simulations
Front impact	15	$6n_i$	90
Side impact	7	$6n_i$	42
Roof crush	11	$5n_i$	55
Modal analysis	12	$4n_i$	48

Many different metamodel types can be used to approximate the detailed FE models (see Section 4.3). First, radial basis function neural networks are chosen. An attractive feature of these metamodels is that they can easily provide an estimation of the prediction error (see Section 4.4.2). One metamodel is built for each response with variables selected according to the screening. The metamodel for the mass is built with all the variables included, i.e. from the sum of the initial datasets from all the loadcases. In order to deter-

mine the DOE sample sizes and judge the quality of the metamodels, the fitting error represented by R^2 and the prediction error represented by $RMSE_{CV}$ are studied. The target is to reach $R^2 > 0.95$ and $RMSE_{CV} < 5\%$ of the mean value. The mass is a linear function of the variables and a perfect fit is therefore obtained. As can be seen in Figure 6.3, the error measure targets are achieved for the responses in the roof crush and the modal analysis loadcases, but not for the other two loadcases. In the hope of improving the accuracy, feedforward neural networks are also fitted to the same datasets. This metamodel type often results in a closer fit to the simulated responses, but it provides no direct information about the prediction error.

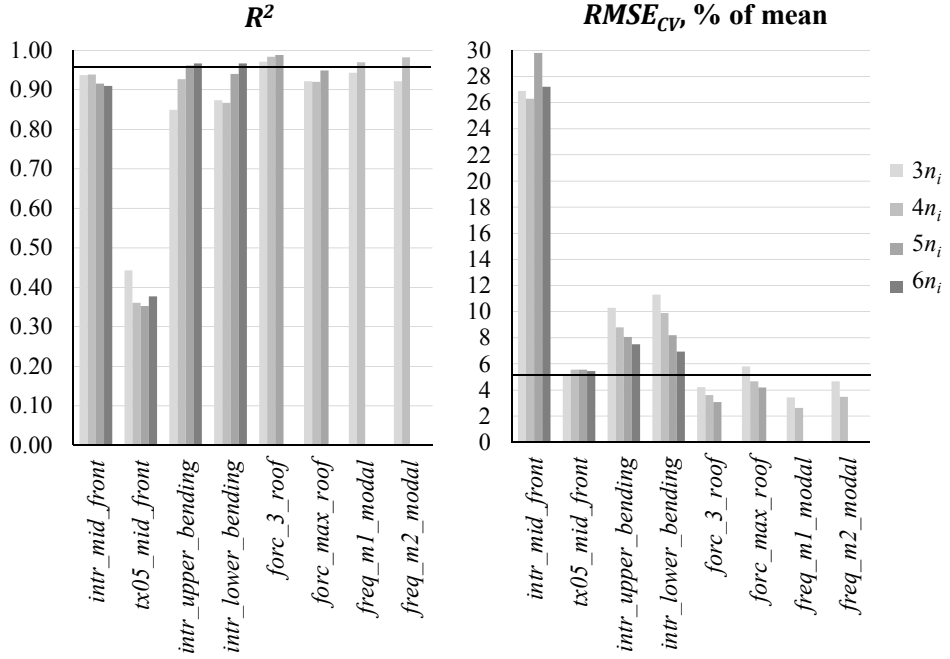

Figure 6.3 Error measures for the RBFNN metamodels representing the constraints.

Several different global optimization algorithms suitable for optimization of complex responses are available (see Section 4.5). In this study, an adaptive simulated annealing algorithm is used for the optimization on both the RBFNN and FFNN metamodels. When the optimization problem is solved with the two sets of metamodels, two different design proposals are found. The design obtained using the FFNN metamodels has a mass reduction of 12.0% of the variable mass, and the design obtained using the RBFNN metamodels has a mass reduction of 7.9%. The design proposals are presented in Table 6.2.

Table 6.2 Design proposals from optimization studies using RBFNN and FFNN metamodels. (+) indicates an increased thickness and (-) a decreased thickness. Results from both metamodels and verification simulations are presented.

Variable	Thickness (mm)				
	Min.	Nom.	Max.	RBFNN	FFNN
t_1001	0.70	1.00	1.30	0.80 (-)	0.85 (-)
t_1002	0.70	1.00	1.30	0.75 (-)	0.85 (-)
t_1003	0.70	1.00	1.30	1.20 (+)	0.80 (-)
t_1004	0.70	1.00	1.30	0.80 (-)	0.70 (-)
t_1005	0.70	1.00	1.30	0.70 (-)	0.70 (-)
t_1006	0.70	1.00	1.30	0.75 (-)	1.05 (+)
t_1007	0.70	1.00	1.30	0.70 (-)	0.70 (-)
t_1008	0.70	1.00	1.30	1.15 (+)	0.70 (-)
t_2010	1.40	2.00	2.60	2.60 (+)	2.25 (+)
t_2011	1.40	2.00	2.60	1.40 (-)	1.55 (-)
t_2012	1.40	2.00	2.60	1.75 (-)	2.25 (+)
t_2013	1.40	2.00	2.60	2.20 (+)	2.15 (+)
t_2014	1.90	2.50	3.10	3.05 (+)	1.90 (-)
t_2015	1.40	2.00	2.60	2.20 (+)	2.15 (+)
t_3016	1.40	2.00	2.60	1.40 (-)	1.40 (-)
t_3017	1.90	2.50	3.10	2.95 (+)	2.95 (+)
t_3018	1.40	2.00	2.60	1.40 (-)	2.40 (+)
t_4019	1.40	2.00	2.60	1.40 (-)	1.40 (-)
t_4020	1.40	2.00	2.60	1.40 (-)	1.40 (-)
t_4021	1.40	2.00	2.60	1.40 (-)	1.40 (-)
t_4022	1.40	2.00	2.60	2.00 (-)	2.40 (+)
t_4023	1.40	2.00	2.60	1.80 (-)	2.15 (+)
t_4024	1.40	2.00	2.60	2.60 (+)	1.45 (-)
t_4025	1.40	2.00	2.60	2.55 (+)	2.50 (+)
t_4026	1.40	2.00	2.60	2.60 (+)	2.55 (+)

Response	Unit	Nom.	Req.	RBFNN		FFNN	
				Pred.	Calc.	Pred.	Calc.
Mass	kg	56.96	Min.	56.30	56.30	55.95	55.95
intr_mid_front	mm	7.74	< 7.8	7.80	5.37	7.77	7.33
tx05_mid_front	ms	11.54	> 11.5	11.51	**11.33**	11.51	11.68
intr_upper_side	mm	25.38	< 25.4	25.11	23.54	25.04	24.28
intr_lower_side	mm	25.37	< 25.4	24.40	23.83	25.24	24.12
forc_3_roof	kN	44.81	> 44.8	49.77	48.78	48.63	46.88
forc_max_roof	kN	57.11	> 57.1	57.34	57.86	57.16	58.86
freq_m1_modal	Hz	107.5	> 107	113.8	115.7	115.3	109.3
freq_m2_modal	Hz	146.5	> 146	146.4	152.7	146.1	148.6
Mass reduction				7.89%		12.01%	

The accuracy of the two sets of metamodels cannot be compared since there is no known prediction error for the FFNN metamodels. Both design proposals are therefore simulated using the detailed FE models. It is found that the design with the largest mass reduction fulfils all the constraints, while the other design violates one of the constraints, see Table 6.2. By studying the deformations and the responses as function of time, it is concluded that no undesired behaviour is obtained for the design with 12.0% mass reduction. The optimization study can thus be considered successful and the design proposal obtained with the FFNN metamodels resulting in a 12.0% mass reduction is seen as the final result. No manual adjustments of the design proposal are necessary since all the constraints are fulfilled. The satisfactory results indicate that no important variables are omitted in the screening process and that the metamodels are reasonably accurate.

7 Discussion

Metamodels have been used for several years to approximate detailed and computationally expensive simulation models. In particular, it has been an attractive method in optimization studies where many evaluations of different variable settings are required. Historically, low-order polynomials, which have the ability to capture the behaviour of a limited part of the design space, have often been used in an iterative approach. Today, the trend goes towards using more advanced metamodels that can capture the behaviour of the complete design space. If the metamodels are sufficiently accurate, the optimization can then be performed in a single-stage approach. For these optimization problems, a stochastic optimization algorithm is often the best choice in order to find the global optimum. The approach with global metamodels is very attractive since it is flexible and can be used for different kind of optimization studies. There is in principle no difference to perform multidisciplinary design optimization compared to perform optimization taking into account responses from only one single loadcase.

Metamodel-based MDO has been investigated by the automotive industry, but has not yet been implemented within the product development process. According to Duddeck (2008), the difficulties so far have mainly been related to insufficient computational resources and inadequate metamodel accuracy. However, new metamodels are being developed and computer capacity is constantly increasing. Therefore, Agte et al. (2010) claim that the late introduction of MDO is more related to organizational and cultural issues than to technical barriers. An MDO process as straight-forward as the one proposed in this thesis is therefore appealing, since it works well in an existing organizational structure and leaves much of the work and part of the decision freedom to the disciplinary experts.

The success of the defined process depends on the accuracy of the metamodels. Some responses from the detailed simulation models are complex and difficult to capture, which can result in inaccurate metamodels. As shown by the presented application example, this is often the case for responses in front crash loadcases, where phenomena related to axial buckling can be hard to model. In situations with limited metamodel accuracy, it is often useful to try different metamodels and optimization algorithms in order to obtain several different design proposals to choose from. Furthermore, it is shown by the presented application example that the results can be satisfactory even when the metamodels do not have the desired accuracy. In addition, if constraints are found to be violated in the

verification simulations, it is often possible to adjust the results manually to obtain an acceptable solution. Although the optimal design will not be obtained in this way, the knowledge gained during the process enables balanced design decisions that can improve the product performance.

By studying the number of detailed simulations that are required to find the new design proposal for the described application example, it can also be concluded that the proposed method is computationally efficient. The total number of simulations with the detailed model for the four different loadcases is 347, distributed among 104 for the screening, 235 for the DOEs, and 8 for the verification. In comparison to the number of simulations required to perform direct optimization or to use a sequential response surface method, this is a low number. As an example, a genetic algorithm with a population of 30 individuals for each of the four loadcases would cover only three generations, if a similar number of simulations as in the described metamodel-based process should be performed. This number is normally too low for the algorithm to find a global optimum. If an SRSM with linear metamodels is used, the number of iterations that can be covered with the same simulation budget is also very limited and probably too small for the process to converge. It should also be noted that the direct optimization and SRSM do not have the same benefits of readily fitting into the organization of a large company. These methods require instead a separate group to perform the complete optimization study.

8 Conclusion and Outlook

The aim of this thesis has been to find a suitable method for implementing multidisciplinary design optimization in automotive development. The goal has been to develop an efficient MDO process for large-scale structural applications where different groups need to work concurrently and autonomously using computationally expensive simulation models. The presented process can be categorized as a single-level MDO method that uses global metamodels to achieve autonomy and concurrency for the groups during the execution of the most computationally demanding steps of the process. The process has been demonstrated in a simple application example and the results show that:

- The presented process is efficient, flexible, and suitable for common structural MDO applications in the automotive industry, such as weight optimization with respect to NVH and crashworthiness.

- The accuracy of metamodels is important, but useful results can also be obtained for metamodels with limited accuracy.

- Although an optimum solution cannot be guaranteed, the method enables balanced design decisions that improve the product performance.

- Several different design proposals can be obtained with a minor additional effort.

- The process can easily fit into an existing organization and product development process with different groups developing the metamodels in parallel using the tools of their choice.

- The different groups of disciplinary experts can take part in the design decisions by participating in the setup of the problem and in the selection of final designs.

A traditional approach during automotive development has been to find a design that meets all requirements, i.e. a design that is feasible but not optimal. By incorporating the described metamodel-based process for multidisciplinary design optimization into the

product development process, there is a potential for designing better products in a shorter time.

Even if the process is suitable for implementation today, there are several aspects that can be studied further. For example would it be interesting to investigate the implications of coupled disciplines, i.e. the existence of coupling variables, to broaden the scope of application. The method is developed to be flexible and have the possibility to incorporate robustness considerations and multiple objectives. The use of the process for different types of problems and applications is therefore welcomed.

The success of the proposed method is relying on global metamodels that can represent the complete design space accurately. It is therefore appropriate also to investigate ways to achieve improved metamodels for complex responses that include e.g. discontinuities.

9 Review of Appended Papers

Paper I

Multidisciplinary design optimization methods for automotive structures

The aim of the first paper is to find suitable MDO methods for large-scale automotive structural applications. The paper includes descriptions and a comparison of a number of MDO formulations presented in the literature. The suitability of these methods is then assessed in relation to the special characteristics of automotive structural MDO problems. It is stated that the method should allow the involved groups to work concurrently and autonomously, must allow the use of metamodels, but does not need to handle coupling variables. It is found that a single-level method in combination with the use of metamodels is often the most convenient way of solving MDO problems for automotive structural applications involving computationally expensive simulations.

Paper II

A metamodel-based multidisciplinary design optimization process for automotive structures

The aim of the second paper is to describe an MDO process for automotive structures and consider aspects related to its implementation. The process is developed to fit a normal product development process of a large automotive company and is based on the findings from the first paper. The requirements placed on the process are presented, and a process meeting these requirements are described. The suitability of the process is then demonstrated in a simple application example. It is concluded that the presented process is efficient, flexible, and suitable for common structural MDO applications within the automotive industry. Furthermore, it fits easily into an existing organization and product development process.

Bibliography

Agte, J., de Weck, O., Sobieszczanski-Sobieski, J., Arendsen, P., Morris, A., and Spieck, M. (2010). MDO: Assessment and direction for advancement - An opinion of one international group. *Structural and Multidisciplinary Optimization, 40*(1-6), 17-33.

Akaike, H. (1970). Statistical predictor identification. *Annals of the Institute of Statistical Mathematics, 22*(1), 203-217.

Alexandrov, N. M., and Lewis, R. M. (2002). Analytical and computational aspects of collaborative optimization for multidisciplinary design. *AIAA Journal, 40*(2), 301-309.

Allison, J., Kokkolaras, M., and Papalambros, P. (2005). On the impact of coupling strength on complex system optimization for single-level formulations. *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference.* Long Beach, California, USA.

Balling, R. J., and Sobieszczanski-Sobieski, J. (1994). Optimization of coupled systems: A critical overview of approaches. AIAA-94-4330-CP. *5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* Panama City Beach, Florida, USA.

Beyer, H.-G., and Schwefel, H.-P. (2002). Evolution strategies - A comprehensive introduction. *Natural Computing, 1*(1), 3-52.

Braun, R. D. (1996). *Collaborative optimization: An architecture for large-scale distributed design.* Ph.D. thesis, Department of Aeronautics and Astronautics, Stanford University.

Breitkopf, P., Naceur, H., Rassineux, A., and Villon, P. (2005). Moving least squares response surface approximation: Formulation and metal forming applications. *Computers and Structures, 83*(17-18), 1411-1428.

Chester, D. L. (1990). Why two hidden layers are better than one. In M. Caudill (Ed.), *Proceedings of the International Joint Conference on Neural Networks (IJCNN-90-WASH DC)*, (pp. 265-268). Washington DC, USA.

Clarke, S. M., Griebsch, J. H., and Simpson, T. W. (2005). Analysis of support vector regression for approximation of complex engineering analyses. *Journal of Mechanical Design, 127*(6), 1077-1087.

Craig, K. J., Stander, N., Dooge, D. A., and Varadappa, S. (2002). MDO of automotive vehicle for crashworthiness and NVH using response surface methods. AIAA 2002-5607. *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* Atlanta, Georgia, USA.

Cramer, E. J., Dennis Jr., J. E., Frank, P. D., Lewis, R. M., and Shubin, G. R. (1994). Problem formulation for multidisciplinary optimization. *SIAM Journal on Optimization, 4*(4), 754-776.

Craven, P., and Wahba, G. (1979). Smoothing noisy data with spline functions. *Numerische Mathematik, 31*(4), 377-403.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182-197.

DeMiguel, A.-V., and Murray, W. (2000). An analysis of collaborative optimization methods. AIAA-2000-4720. *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 6-8 September 2000.* Long Beach, California, USA.

Duddeck, F. (2008). Multidisciplinary optimization of car bodies. *Structural and Multidisciplinary Optimization, 35*(4), 375-389.

Dyn, N., Levin, D., and Rippa, S. (1986). Numerical procedures for surface fitting of scattered data by radial functions. *SIAM Journal on Scientific and Statistical Computing, 7*(2), 639-659.

Eiben, A. E., and Smith, J. E. (2003). *Introduction to evolutionary computing.* Berlin: Springer.

Fang, K.-T., J., D. K., Winker, P., and Zhang, Y. (2000). Uniform design: Theory and application. *Technometrics, 42*(3), 237-248.

Forrester, A. I., and Keane, A. J. (2009). Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences, 45*(1-3), 50-79.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics, 19*(1), 1-67.

Geman, S., and Geman, D. (1984). Stochastic relaxation, Gibbs distribution, and the Bayesian restoration in images. *IEEE Transactions of Pattern Analysis and Machine Intelligence, 6*(6), 721-741.

Giesing, J. P., and Barthelemy, J. M. (1998). A summary of industry MDO applications and needs. *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* St. Louis, Missouri, USA.

Goel, T., Haftka, R., Shyy, W., and Queipo, N. (2007). Ensemble of surrogates. *Structural and Multidisciplinary Optimization, 33*(3), 199-216.

Goel, T., and Stander, N. (2009). Comparing three error criteria for selecting radial basis function network topology. *Computer Methods in Applied Mechanics and Engineering, 198*(27-29), 2137-2150.

Gu, L., and Yang, R. (2006). On reliability-based optimisation methods for automotive structures. *International Journal of Materials and Product Technology, 24*(1-3), 3-26.

Hardy, R. (1990). Theory and applications of the multiquadric-biharmonic method, 20 years of discovery 1968-1988. *Computers & Mathematics with Applications, 19*(8-9), 163-208.

Holland, J. H. (1992). *Adaptation in natural and artificial systems : An introductory analysis with applications to biology, control, and artificial intelligence.* Cambridge: MIT Press.

Hoppe, A., Kaufmann, M., and Lauber, B. (2005). Multidisciplinary optimization considering crash and NVH loadcases. *ATZ/MTZ Virtual Product Creation.* Stuttgart, Germany.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359-366.

Hu, X., Eberhart, R., and Shi, Y. (2003). Engineering optimization with particle swarm. *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, (pp. 53-57). Indianapolis, Indiana, USA.

Hwang, G.-H., and Jang, W.-T. (2008). An adaptive evolutionary algorithm combining evolution strategy and genetic algorithm (application of fuzzy power system stabilizer). In W. Kosinsky (Ed.), *Advances in Evolutionary Algorithms* (pp. 95-116). Vienna: InTech.

Ingber, L. (1996). Adaptive simulated annealing (ASA): Lessons learned. *Control and Cybernetics, 25*(1), 32-54.

Iooss, B., Boussouf, L., Feuillard, V., and Marrel, A. (2010). Numerical studies of the metamodel fitting and validation processes. *International Journal on Advances in Systems and Measurements, 3*(1 & 2), 11-21.

Jin, R., Chen, W., and Simpson, T. (2001). Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization, 23*(1), 1-13.

Jin, R., Chen, W., and Sudjianto, A. (2002). On sequential sampling for global metamodeling in engineering design. DETC2002/DAC-34092. *Proceedings of the ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2002).* Montreal, Canada.

Johnson, M., Moore, L., and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference, 26*(2), 131-148.

Kalagnanam, J. R., and Diwekar, U. M. (1997). An efficient sampling technique for off-line quality control. *Technometrics, 39*(3), 308-319.

Kennedy, J., and Eberhart, R. (1995). Particle swarm optimization. *Proceedings of the 1995 IEEE International Conference on Neural Networks*, (pp. 1942-1948). Perth, Australia.

Kim, H. M. (2001). *Target cascading in optimal system design.* Ph.D. thesis, Mechanical Engineering, University of Michigan, Ann Arbor.

Kim, B.-S., Lee, Y.-B., and Choi, D.-H. (2009). Comparison study on the accuracy of metamodeling technique for non-convex functions. *Journal of Mechanical Science and Technology, 23*(4), 1175-1181.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671-680.

Kodiyalam, S., and Sobieszczanski-Sobieski, J. (2000). Bilevel integrated system synthesis with response surfaces. *AIAA journal, 38*(8), 1479-1485.

Kodiyalam, S., and Sobieszczanski-Sobieski, J. (2001). Multidisciplinary design optimization - Some formal methods, framework requirements, and application to vehicle design. *International Journal of Vehicle Design, 25*(1-2 SPEC. ISS.), 3-22.

Kodiyalam, S., Yang, R., Gu, L., and Tho, C.-H. (2004). Multidisciplinary design optimization of a vehicle system in a scalable, high performance computing environment. *Structural and Multidisciplinary Optimization, 26*(3), 256-263.

Koehler, J., and Owen, A. (1996). Design and analysis of experiments. In S. Ghosh, and C. Rao (Eds.), *Handbook of Statistics* (pp. 261-308). Amsterdam: North-Holland.

Kroo, I., Altus, S., Braun, R., Gage, P., and Sobieski, I. (1994). Multidisciplinary optimization methods for aircraft preliminary design. AIAA-94-4325-CP. *5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* Panama City Beach, Florida, USA.

Kroo, I., and Manning, V. (2000). Collaborative optimization: status and directions. AIAA-2000-4721. *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* Long Beach, California, USA.

Li, Y., Ng, S., Xie, M., and Goh, T. (2010). A systematic comparison of metamodeling techniques for simulation optimization in decision support systems. *Applied Soft Computing, 10*(4), 1257-1273.

Lin, Y. (2004). *An efficient robust concept exploration method and sequential exploratory experimental design.* Ph.D. thesis, Mechanical Engineering, Georgia Institute of Technology, Atlanta.

Martin, J. D., and Simpson, T. W. (2004). On the use of Kriging models to approximate deterministic computer models. DETC2004/DAC-57300. *Proceedings of the ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2004).* Salt Lake City, Utah, USA.

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics, 21*(2), 239-245.

Meckesheimer, M., Booker, A., Barton, R., and Simpson, T. (2002). Computationally inexpensive metamodel assessment strategies. *AIAA Journal, 40*(10), 2053-2060.

Michalek, J. J., and Papalambros, P. Y. (2005a). An efficient weighting update method to achieve acceptable consistency deviation in analytical target cascading. *Journal of Mechanical Design, Transactions of the ASME, 127*(2), 206-214.

Michalek, J. J., and Papalambros, P. Y. (2005b). Weights, norms, and notation in analytical target cascading. *Journal of Mechanical Design, Transactions of the ASME, 127*(3), 499-501.

Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics, 33*(2), 161-174.

Myers, R. H., Montgomery, D. C., and Andersson-Cook, C. M. (2008). *Response surface methodology: Process and product optimization using designed experiments* (Third ed.). Hoboken, New Jersey, USA: Wiley.

Owen, A. B. (1992). Orthogonal arrays for computer experiments, integration and visualization. *Statistica Sinica, 2*, 439-452.

Pan, J., and Diaz, A. R. (1989). Some results in optimization of non-hierarchic systems. *The 1989 ASME Design Technical Conferences - 15th Design Automation Conference.* Montreal, Quebec, Canada.

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

Renaud, J. E., and Gabriele, G. A. (1991). Sequential global approximation in non-hierarchic system decomposition and optimization. *17th Design Automation Conference presented at the 1991 ASME Design Technical Conferences*, *32*, pp. 191-200. Miami, Florida, USA.

Renaud, J. E., and Gabriele, G. A. (1993). Improved coordination in nonhierarchic system optimization. *AIAA journal, 31*(12), 2367-2373.

Renaud, J. E., and Gabriele, G. A. (1994). Approximation in nonhierarchic system optimization. *AIAA journal, 32*(1), 198-205.

Roth, B. D. (2008). *Aircraft family design using enhanced collaborative optimization.* Ph.D. thesis, Department of Aeronautics and Astronautics, Stanford University.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal represenations by error propagation. In D. E. Rumelhart, and J. L. McClelland (Eds.), *Parallel distributed processing: explorations in the microstructure of cognition* (Vol. 1: Foundations, pp. 318-362). Cambridge: MIT Press.

Ryberg, A.-B., Bäckryd, R. D., and Nilsson, L. (2012). *Metamodel-based multidisciplinary design optimization for automotive applications.* Technical Report LIU-IEI-R-12/003. Linköping University, Division of Solid Mechanics.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science, 4*(4), 409-423.

Shankar, J., Haftka, R. T., and Watson, L. T. (1993). Computational study of a nonhierarchical decomposition algorithm. *Computational Optimization and Applications, 2*(3), 273-293.

Sheldon, A., Helwig, E., and Cho, Y.-B. (2011). Investigation and application of multi-disciplinary optimization for automotive body-in-white development. *Proceedings of the 8th European LS-DYNA Users Conference.* Strasbourg, France.

Shi, L., Yang, R. J., and Zhu, P. (2012). A method for selecting surrogate models in crashworthiness optimization. *Structural and Multidisciplinary Optimization, 46*(2), 159-170.

Simpson, T., Peplinski, J., Koch, P. N., and Allen, J. (2001). Metamodels for computer-based engineering design: Survey and recommendations. *Engineering with Computers, 17*(2), 129-150.

Smola, A. J., and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing, 14*(3), 199-222.

Sobieszczanski-Sobieski, J. (1988). Optimization by decomposition: A step from hierarchic to non-hierarchic systems. *Second NASA/Air Force Symposium on Recent Advances in Multidisciplinary Analysis and Optimization, September 1988.* Hampton, Virginia, USA.

Sobieszczanski-Sobieski, J., Agte, J. S., and Sandusky Jr., R. R. (1998). Bi-level integrated system synthesis (BLISS). *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* St. Louis, Missouri, USA.

Sobieszczanski-Sobieski, J., Altus, T. D., Phillips, M., and Sandusky, R. (2003). Bilevel integrated system synthesis for concurrent and distributed processing. *AIAA Journal, 41*(10), 1996-2003.

Sobieszczanski-Sobieski, J., Kodiyalam, S., and Yang, R. (2001). Optimization of car body under constraints of noise, vibration, and harshness (NVH), and crash. *Structural and Multidisciplinary Optimization, 22*(4), 295-306.

Sobieski, I. P., and Kroo, I. M. (2000). Collaborative optimization using response surface estimation. *AIAA journal, 38*(10), 1931-1938.

Sobol', I. M. (2001). Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation, 55*(1-3), 271-280.

Stander, N., Roux, W., Goel, T., Eggleston, T., and Craig, K. (2010). *LS-Opt user's manual version 4.1.* Livermore: Livermore Software Technology Corporation.

Swiler, L. P., and Wyss, G. D. (2004). *A user's guide to Sandia's latin hypercube sampling software: LHS UNIX library/standalone version.* Technical Report SAND2004-2439, Sandia National Laboratories, Albuquerque, New Mexico, USA.

Tang, B. (1993). Orthogonal array-based latin hypercubes. *Journal of the American Statistical Associasion, 88*(424), 1392-1397.

Tosserams, S., Etman, L. F., Papalambros, P. Y., and Rooda, J. E. (2006). An augmented lagrangian relaxation for analytical target cascading using the alternating direction method of multipliers. *Structural and Multidisciplinary Optimization, 31*(3), 176-189.

Venter, G. (2010). Review of optimization techniques. In R. Blockley, and W. Shyy (Eds.), *Encyclopedia of aerospace engineering* (Vol. 8: System Engineering). Chichester, West Sussex, UK: John Wiley & Sons.

Viana, F. A., Gogu, C., and Haftka, R. T. (2010). Making the most out of surrogate models: tricks of the trade. DETC2010-28813. *Proceedings of the ASME 2010 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, (pp. 587-598). Montreal, Quebec, Canada.

Wang, G. G., and Shan, S. (2007). Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design, 129*(4), 370-380.

Yang, R. J., Gu, L., Tho, C. H., and Sobieszczanski-Sobieski, J. (2001). Multidisciplinary design optimization of a full vehicle with high performance computing, AIAA-2001-1273. *Proceedings of the 42nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit.* Seattle, Washington, USA.

Younis, A., and Dong, Z. (2010). Trends, features, and tests of common and recently introduced global optimization methods. *Engineering Optimization, 42*(8), 691-718.

Zerpa, L. E., Queipo, N. V., Pintos, S., and Salager, J.-L. (2005). An optimization methodology of alkaline-surfactant-polymer flooding processes using field scale numerical simulation and multiple surrogates. *Journal of Petroleum Science and Engineering, 47*(3-4), 197-208.

Zhao, D., and Xue, D. (2011). A multi-surrogate approximation method for metamodeling. *Engineering with Computers, 27*(2), 139-153.