Adaptieve surrogaatmodellering voor simulatiegebaseerd ontwerp

Grid-Enabled Adaptive Surrogate Modeling for Computer Aided Engineering

Dirk Gorissen

Promotoren: prof. dr. ir. T. Dhaene, prof. dr. J. Broeckhove
Proefschrift ingediend tot het behalen van de graden van
Doctor in de Ingenieurswetenschappen: Computerwetenschappen (UGent) en
Doctor in de Wetenschappen: Informatica (UA)

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. D. De Zutter
Faculteit Ingenieurswetenschappen

Departement Wiskunde-Informatica
Voorzitter: prof. dr. G. Van Steen
Faculteit Wetenschappen

Academiejaar 2009 - 2010

UNIVERSITEIT GENT

Universiteit Antwerpen

Acknowledgments

Knowledge is in the end based on acknowledgment.
–Ludwig Wittgenstein

After 300 odd pages the text for this dissertation is almost complete. These 300 pages have taken quite some time and effort to write and are the result of the four years of work that constitutes a PhD. The 60 cm worth of papers sitting on my desk beside me are a nice reminder in that respect. However, without a doubt many, if not all, of these pages would not be filled were it not for the help, guidance, and support from many people.

First and foremost, I would like to thank my professor and main adviser Tom Dhaene for the opportunity to do a PhD and for the inspiration, guidance, patience, and motivation during the PhD process. This has helped me grow both personally and intellectually. I have always very much appreciated the room and flexibility given to develop my PhD research and the 100% support I enjoyed whenever I came up with some eyebrow-raising idea or plan. The permission to follow the Master of Artificial Intelligence program during my first to years being a good example. As do I appreciate the many interesting conferences I have been able to attend thanks to Tom's consent and financial support. Secondly I want to thank Prof. Piet Demeester for accommodating me in the IBCN group and for always being generally helpful and supportive.

Thirdly, I want to specially thank my second co-adviser Prof. Jan Broeckhove from the University of Antwerp. Foremost for being the one to first give me the opportunity to continue my Masters' thesis on to the PhD level and for having given me the unique opportunity to work at Emory University, Atlanta, for half a year. Furthermore, the fact that I have been able to continue to occupy an office at the University of Antwerp, even after my official move to Ghent University, is something that I really appreciate, as is the help I have always readily received. In this spirit thanks also to Eric Laermans for the interesting discussions and feedback on papers, ideas, and on this dissertation. For the same reason I wish to thank Prof. Jack P. C. Kleijnen from Tilburg University for being part of my reading committee and for the many valuable corrections and comments. They really have improved the manuscript. Thanks also to the other committee members: Prof. Daniel De Zutter, Prof. Frans Arickx and Prof. Luc Dupré. I also want to give special thanks to Prof. Qi-Jun Zhang form Carleton University, Ottawa. First of all for accepting me as a research visitor in his department and secondly for the enthusiasm and patience he always showed during my memorable stay there. It was an experience I will not forget and it taught me a lot about conducting solid research.

My gratitude also goes out through the many colleagues I have had the privilege of working with over the past five years. Ivo Couckuyt from Ghent and Karel Crombecq & Wouter Hendrickx from Antwerp for the many interesting, critical, and challenging discussions we had over the years. It has been a pleasure being part of the team and I think we can be proud of what we achieved together. To Luciano De Tomassi, it was nice working with you, both on a personal and professional level. Good luck with your further career. Thanks also to Kurt Vanmechelen for the many sharp comments and advice on so many domains. The same to Gunther Stuer. If it wasn't for your encouragement I would never have started with the PhD track in the first place. To all my colleagues, I always enjoyed sharing an office with you guys, both professionally and personally. I wish you all well on your further careers. From Antwerp: Sam Verboven, Ruben Van den Bossche, Peter Hellinckx, and others. From Ghent: Dirk Deschrijver, Jeroen Famaey, Femke Ongenae, Femke De Backere, Bert Vankeirsbilck, Huu Minh Nguyen and others. From Carleton: Lei Zhang, Yi Cao, and others. From Emory: Vaidy Sunderam and Dawid Kurzyniec. Dawid, I learned more from you during those five months than in the preceding two years together. It was a pleasure working with you and I hope to see you sometime again in Krakow. Thanks also to Dirk De Villiers, my South African compadre and Hamed Rouhani, my Iranian colleague and friend. I wish both of you well and hope to see you in the future somewhere.

I would also like to thank a number of people who provided, data and/or interesting applications for me to work on. Without their willingness to contribute I wouldn't have had half the case I have today. In this respect I want to thank Markus Ganser and Karen Grossenbacher from BMW motor company for making the automotive data available and the fruitful discussions. Also thank you to Jeroen Croon and Joost Rommes from the NXP-TSMC Research Center, Eindhoven for the LNA simulation code. Thank you to Matthias Ihme from Stanford University for providing the chemistry combustion data, and Robert Gramacy from the University of California, Santa Cruz for providing the data on the LGBB example. Special thanks also to Robert Lehmensiek and Petrie Meyer from the University of Stellenbosch, South Africa. For providing the simulation code for the Step discontinuity/Inductive posts problems and for being kind enough to host me in Stellenbosch for a few weeks.

On a personal side I am forever indebted to my fiancée Anne. I love you very much and its hard to thank you enough for your love, support, understanding, and patience. It wasn't always easy having to put up with me and my computer and the different university trips. Always, caring, supportive and best cook I know of. Thank you.

I also owe great thanks to my family for their continued support and encouragement. Even if nobody ever really understood what I was doing. I haven't always been home as much as I wanted, yet I am very lucky to have you all. I also would like to thank all my friends that helped me and spent good times with me in the past years. Laurent Vergauwen, we go way back and I'm privileged to call you my friend. Peter "Big Pete" Thompson, as reliable as a Malawian bus service but I miss you man, hope to see you soon. Hanna, there is not much I can say here, thank you and take care. The van Dijck family, for many years you served as my second home, you are a model to us all. To Hans Schippers, Ronald Verboven, Kristof Boeynaems, and Allan De Brueker a well meant "Santé!".

Finally, to all the others that have helped me over the years, or those whose paths I crossed while abroad: if I have failed to mention you due to my own ignorance, my apologies. Know that your support and friendship is and has been much appreciated.

Dirk Gorissen
Gent, Belgium
March 2010

Table of Contents

Nederlandse samenvatting –Summary in Dutch–

Bij veel problemen uit de (ingenieurs-) wetenschappen is het moeilijk of zelfs onmogelijk om fysische experimenten rechtstreeks uit te voeren. In plaats daarvan worden complexe simulatieprogramma's gebruikt om virtuele experimenten uit te voeren op krachtige computers of clusters. Hoewel dat wetenschappers en ingenieurs meer flexibiliteit biedt om designproblemen en -fenomenen te bestuderen onder gecontroleerde omstandigheden, vereisen computersimulaties een aanzienlijke investering in tijd en rekenkracht. De duur van één enkele simulatie kan variëren van enkele minuten of uren tot enkele dagen of zelfs weken. Bijgevolg worden gangbare technieken zoals optimalisatie, sensitiviteitsanalyse, en ontwerpruimte-exploratie al snel zeer moeilijk tot onmogelijk.

Om hier mee om te gaan doen onderzoekers beroep op verschillende approximatie methoden die het gedrag van de simulatiecode zo goed mogelijk benaderen maar tegelijkertijd veel goedkoper zijn om te evalueren. Deze thesis concentreert zich op datagebaseerde, globale surrogaatmodellen als approximatiemethode. Het doel van globale surrogaatmodellering is om een globaal approximatiemodel te construeren dat zo accuraat mogelijk is, met zo weinig mogelijk simulaties, en op een zo efficiënt mogelijke manier. Eens geconstrueerd, kan een dergelijk surrogaatmodel hergebruikt worden in de verdere stappen van het design proces of geïntegreerd worden in software pakketten. De besproken problematiek doet zich voor in een breed scala van domeinen, gaande van aërodynamica tot hydrologie.

Echter, de constructie van zulke globale surrogaatmodellen vereist de resolutie van een heel aantal problemen en keuzes. Deze omvatten bijvoorbeeld de keuze van model type, modelselectiecriteria, model parameter optimalizatie algoritme, etc.. In de praktijk blijkt dat een designer deze keuzes op een vrij pragmatische en ad hoc manier maakt en dat er weinig geïntegreerde tools zijn om op een systematische en adaptieve manier surrogaatmodellen te construeren. Er is ruimte voor een meer flexibele, uitbreidbare, en adaptieve benadering voor het modeleringsprobleem dat geen harde veronderstellingen vereist (maar ze ook niet negeert) over de karakteristieken van het probleemdomein of modeleringsaanpak. Tezelfdertijd moet men rekening houden met het feit dat het primair doel van een domeinexpert ligt in het oplossen van zijn designprobleem. Een accuraat, globaal surrogaatmodel is hierbij belangrijk maar geen doel op zich. De tijd en leercurve die een domeinexpert hierin moet investeren moet dan ook geminimaliseerd worden.

Deze observatie vormt het uitgangspunt van deze thesis. De thesis onderzoekt hoe de adoptiedrempel van geavanceerde surrogaatmodeleringstechnieken kan verkleind worden zodat deze gemakkelijker gebruikt kunnen worden door domeinexperts om hun designresultaten te verbeteren en tijdswinst te creëren. Het domein van deze thesis bevindt zich in de doorsnede van machine learning/AI, gedistribueerde systemen, computersimulatie, en software engineering. Centraal staat het ontwerp en de implementatie van een software platform dat verschillende surrogaatmodeleringstechnieken integreert in een flexibele implementatie met enkele extra uitbreidingen. Dit software platform zal nuttig blijken in elk domein waar een goedkoop, accuraat, approximatiemodel nodig is voor een dure simulator of andere databron.

English summary

For many problems from science, and engineering it is impractical to perform experiments on the physical world directly. Instead, complex, physics-based simulation codes are used to run experiments on computer hardware. While allowing scientists more flexibility to study phenomena under controlled conditions, computer experiments require a substantial investment of computation time. One model evaluation may take many minutes, hours, days or even weeks. This is especially problematic for routine tasks such as optimization, sensitivity analysis and design space exploration.

As a result researchers have turned to various approximation methods that mimic the behavior of the simulation model as closely as possible while being computationally cheap(er) to evaluate. This work concentrates on the use of data-driven, global approximations using compact surrogate models. The goal of global surrogate modeling is the generation of a surrogate that is as accurate as possible, using as few simulation evaluations as possible, and with as little overhead as possible. Once such a simpler approximation is available, it can be reused further down the engineering design pipeline. This type of problem and use case is encountered in a very large range of scientific disciplines and fields, ranging from mechanical engineering to hydrology.

However, there are an overwhelming number of options available to the designer: different model types, different experimental designs, different model selection criteria, different hyperparameter optimization strategies, etc. However, in practice it turns out that the designer rarely tries out more than one subset of options and that readily available algorithms and tools to tackle this problem are scarce and limited (particularly in industry). There is room for an extensible, flexible, and automated approach to surrogate modeling, that does not mandate assumptions (but does not preclude them either) about the problem, model type, sampling algorithm, etc. At the same time, the primary concern of a domain expert scientist is obtaining an accurate replacement metamodel for their problem as fast as possible and with minimal user interaction. The surrogate modeling specifics are of lesser or no interest to them.

This thesis starts form this observation and investigates how the barrier of entry for a domain expert can be reduced when it comes to applying state-of-the-art surrogate modeling techniques to his/her application domain. This involves working in the intersection of machine learning/AI, distributed systems, modeling & simulation, and software engineering. Core in this dissertation is the design and implementation of a software platform that brings together many surrogate modeling methods in a flexible implementation with a number of custom extensions and added integration. The framework will be useful in any domain where a cheap, accurate approximation is needed for an expensive reference model and help bridge the gap between theory and application.

Abbreviations

ADS : Advanced Design System
AIC : Aikaike Information Criterion
ANN : Artificial Neural Network
BEEQ : Bayesian Error Estimation Quotient
BIC : Bayesian Information Criterion
CGA : Cannonical Genetic Algorithm
CS : Classifier Systems
DGA : Distributed Genetic Algorithm
DIRECT : DIviding RECTangles
DM : Difference Mapping
EA : Evolutionary Algorithm
EGO : Efficient Global Optimization
EI : Expected Improvement
EM : Electro-Magnetics
EP : Extinction Prevention
GA : Genetic Algorithm
GIS : Geographic Information System
GP : Genetic Programming / Gaussian Process
IM : Input Mapping
KBNN : Knowledge Based Neural Networks
LHS : Latin Hypercube Sampling
LNA : Low Noise Amplifier
LOLA : Local Linear Approximation
LRM : Linear Reference Model
LS-SVM : Least Squares Support Vector Machine
LTI : Linear Time Invariant
LWR : Locally Weighted Regression
MAO : Metamodel Assisted Optimization
MC : Monte Carlo
MCMC : Multi Chain Monte Carlo
MOR : Model Order Reduction
MOSBO : Multi-Objective Surrogate Based Optimization

MPI	Message Passing Interface
MPS	Mode Persuing Sampling
MSE	Mean Squared Error
NIC	Network Information Criterion
OM	Output Mapping
PDF	Probability Distribution Function
PGA	Parallel Genetic Algorithm
PKI	Prior Knowledge Input
PMOR	Parameterized Model Order Reduction
POD	Proper Orthogonal Decomposition
PSE	Problem Solving Environment
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RBFNN	Radial Basis Function Neural Network
ROM	Reduced Order Modeling
RSM	Response Surface Methodology
SBO	Surrogate Based Optimization
SGE	Sun Grid Engine
SM	Space Mapping
SOA	Service Oriented Architecture
SQM	Sample Queue Manager
SQP	Sequential Quadratic Programming
SUMO	Surrogate Modeling
SVM	Support Vector Machine
VC	Vapnik-Chervonenkis
VF	Vector Fitting
XML	Extensible Markup Language

Preamble

All of science can be divided into physics and stamp-collecting.

– Lord Kelvin

The first words are always hardest to write down and be content with. An easy way out is to start from a quote, preferably from somebody of high stature or fame. In that spirit, this introductory preamble reminds me of something a well known Aikido instructor once said: the success of technique is not determined by performing the movement itself but by the way you enter into it and leave from it. It is all too easy to get caught up in the movement itself and forget that what really counts is the control, posture, awareness, discipline, and confidence you show on entering and leaving. Consequently, these are the hardest to master.

Writing papers or a thesis is in many ways similar. The technical side of things writes itself, but the text leading up to and winding down from the technical discussion is much trickier given the need for continuity, coherence, and at least some form of eloquence and parsimony. Inspired (and humbled) by such lucid writers as Richard Dawkins, this preamble is one such attempt. The topic of this thesis is firmly rooted in computer science but influenced by the wide range of scientific fields that have held my interest over the years. I admit to having regarded scientists in other fields with some envy. They were out in expensive labs or adventurous expeditions uncovering the real unknowns of this world. While we computer scientists sit under fluorescent lighting, locked to a fully deterministic and predictable machine, and faced with a somewhat odd gender ratio. However, the truth is that many (if not most) of recent scientific breakthroughs in physics, biology, medicine, and other fields would not have happened without the algorithmic and architecture work of computer scientists like Babbage, Turing, Amdahl, Berners-Lee, and many others. The dichotomy is not as strict as I present it here (the work by Goedel comes to mind). But this can conveniently be ignored for now. Moreover, while the computer scientist may be locked to his reverberating machine, at the same time he enjoys unlimited freedom. His work and creations limited only by human creativity and ingenuity and not by the price of daedalean looking equipment, expensive expeditions, or the need for a library or archive. Furthermore, while it was the eminent Lord Kelvin who said *All of science can be divided into physics and stamp-collecting.* It was also Lord Kelvin who said *I have not the smallest molecule of faith in aerial navigation other than ballooning or of* *expectation of good results from any of the trials we hear of.*

While computer science itself is a very young field, its roots go further back than one might think at first. The word *algorithm* derives from Al-Khwarizmi, a Persian mathematician who wrote a book around the year 825, *On the Calculation with Hindu Numerals*, that was principally responsible for the diffusion of the Indian system of numeration in the Middle East and then Europe. This book was translated into Latin around the 12th century: *Algoritmi de numero Indorum*, and marked the start of the concept algorithm as we know it today. It took until 1642 for this work to materialize in a tangible, working machine. It is that year that the first mechanical adding machine (the *Pascaline*) was developed by the famous French mathematician Blaise Pascal. This was followed by the first 4-function calculator which saw the light around Darwin's time in 1893. The real revolution started in 1943 when Alan Turing and others Completed *Colossus* (the first all electronic calculating device), followed by the discovery of the transistor at Bell Labs 4 years later. The rest, such as the formulation of Moore's Law in 1965, is as they say, history.

The result of this long history and the increasing pace and scale of scientific research is that the days of such Homo Universali as Da Vinci, Al-Khwarizmi, Zhan Heng, and Goethe are no more. This quickly becomes obvious as one starts delving into the topic of ones own research. One soon realizes that Aristotle had it right when he said that *"the more you know, the more you know you don't know"*. There are so many fields and sub-fields, and sub-sub-fields, that one sees no other choice but to take refuge in that safe, but infamous, ivory tower. The work presented in this thesis is no exception, covering only a very narrow segment of that feat known as scientific research. But luckily the topic was such that it allowed the opening of a window here or there. Even if just for a peek.

1 Introduction

Begin at the beginning and go on until you come to the end; then stop.

– Said by the king to the white rabbit in Alice in Wonderland

A thousand years ago science revolved around the description of natural phenomena. Slowly evolving from more esoteric and religious explanations to more scientific explanations involving fundamental concepts such as prediction and replication. The last few hundred years work concentrated on the theoretical fundamentals of science. Marked by astonishing genius and meticulous experimentation, many of the corner stones of modern science and its methods date from this time (Bacon, Galilei, Newton, Maxwell, Planck, etc.). The last few decades science has moved more and more into the computational domain, with technological advances vastly increasing the realm and complexity of problems and phenomena that can be studied. This has led to an explosion of data and tools that are at present being unified into, what is often referred to as, e-Science [1]: the unification of theory, experiment, simulation, and knowledge management. Against this backdrop we can define the problem domain and scope of this dissertation.

1.1 Problem domain and scope

An illustration of the context of this work is given in figure 1.1. A detailed overview of the problem domain in this thesis will be given in chapter 2. In this section a global overview suffices

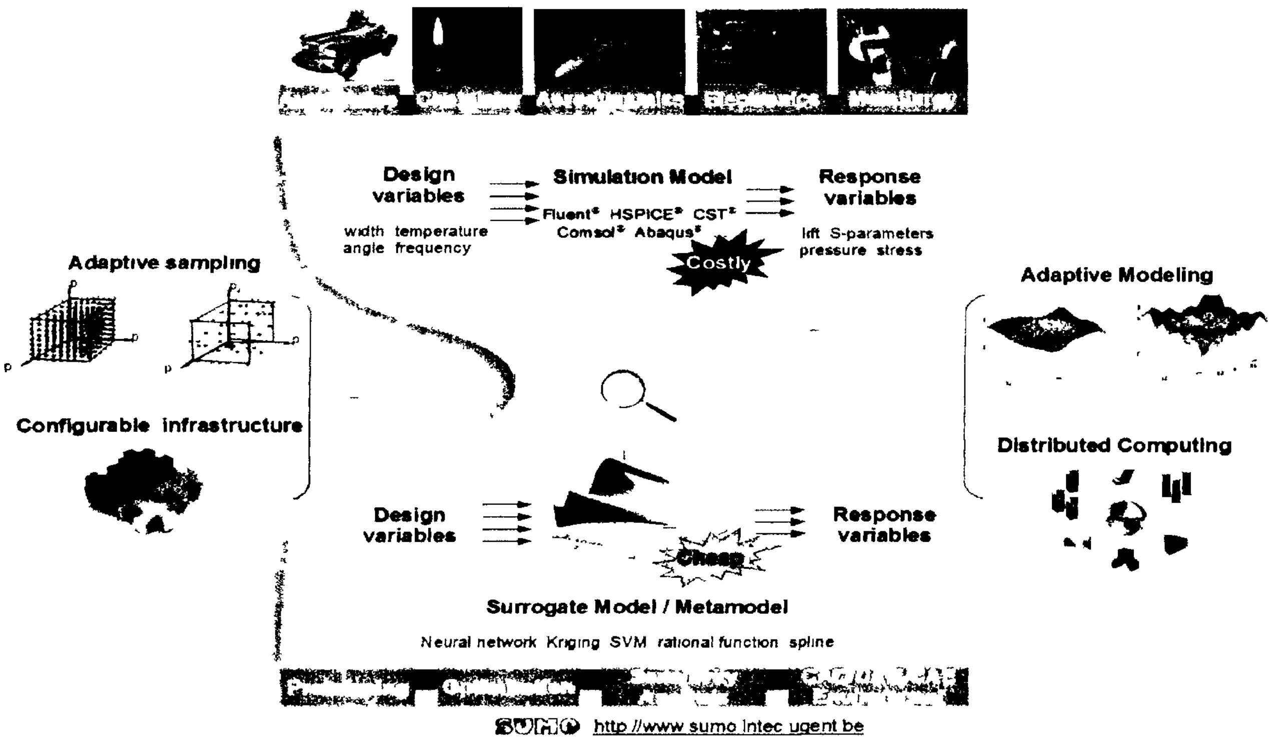

Design variables
Simulation Model
Response variables
Fluent* HSPICE* CST*
Comsol* Abaqus*
width temperature angle frequency
lift S-parameters pressure stress
Costly
Adaptive sampling
Adaptive Modeling
Configurable infrastructure
Distributed Computing
Design variables
Response variables
Cheap
Surrogate Model / Metamodel
Neural network Kriging SVM rational function spline
http://www.sumo.intec.ugent.be

Figure 1.1: Thesis problem domain

1.1.1 Background

Every scientific and engineering field is confronted with physical phenomena that require explanation or reproduction. In this respect computer simulation has been a major vehicle for searching and testing satisfactory theories and solutions. Namely, for many problems it is impractical to perform experiments on the physical world directly (e.g., airfoil design, earthquake propagation). Instead, complex, physics-based simulation codes are used to run experiments on computer hardware.

However, while allowing scientists more flexibility to study phenomena under controlled conditions, computer experiments require a substantial investment of computation time. Engineers are confronted with large design spaces and many variables whose relationship needs to be analyzed. Even if a single simulation takes only a few seconds, routine tasks such as optimization, sensitivity analysis, design space exploration and visualization quickly become cumbersome and impractical. A classical illustrative quote is one by Wang and Shan [2]:

> *...it is reported that it takes Ford Motor Company about 36-160 hrs to run one crash simulation [3]. For a two-variable optimization problem, assuming on average 50 iterations are needed by optimization and assuming each iteration needs one crash simulation, the total computation time would be 75 days to 11 months, which is unacceptable in practice.*

Moore's law provides some solace but is not sufficient to offset the drive to finer timescales, higher resolution, and the known complexity of the algorithms that lie at the heart of simulation codes.

This thesis is concerned with how this problem is often tackled in engineering design: simpler approximation models are created to predict the system performance and develop a relationship between the system inputs and outputs. When properly constructed, these approximation models mimic the behavior of the simulation code while being computationally cheap(er) to evaluate. Different approximation methods exist, each with their relative merits. This work concentrates on the use of data-driven, ***global*** approximations using compact surrogate models (also known as emulators, metamodels, replacement models, or response surface models). The word *global* is crucial here. We are interested in capturing the global behavior of the system in an accurate, efficient model. Not in 'simply' optimizing it. In contrast, "*...in design optimization one is not concerned with the accuracy of any intermediate predictions, but only with the ability to ultimately reach good quality designs.*" [4]. Once such a global approximation is available it is of great use for gaining insight into the behavior of the underlying system. The model may be easily queried, optimized, visualized, and seamlessly integrated into CAD/CAE software packages.

1.1.2 Scope

The operating context of this thesis is thus the problem of generating an accurate data-based global model for an expensive reference model at a minimum computational cost. Thus this thesis will not focus on the reference model itself (what sub-systems it contains, how it is validated, etc.) or how metamodels for different sub-systems may be integrated and used. Or put otherwise, in the terminology of Meckesheimer [5], we only consider the metamodeling module and not the system integration or model formulation modules.

Furthermore, since we cannot possibly provide a discussion for every possible system we must restrict ourselves to a particular class of computer simulation codes. We make the following three fundamental assumptions:

1. We only consider static input-output systems where the output of the simulator at time t depends only on the input at time t. Of course the implementation of the simulator may itself include dynamics and time based behavior. However, we assume that such dynamics are internal to the system, and that the external behavior can be represented by a multivariate function $f : \mathbb{R}^d \mapsto \mathbb{C}^q$. Systems where this function is time dependent or involves prediction of future states (time series prediction) are not considered.

2. Data is generated from computer simulations and is thus deterministic and (quasi) noise free. Some numerical or discretization noise may be present but the results

themselves are deterministic (versus stochastic simulation). This means that the statistical theory built up for the analysis of physical experiments does not apply. Or, as Sacks et. al. [6] put it "*the classical notions of experimental blocking, replication and randomization are irrelevant*" [1].

3. Data is expensive to generate thus the number of simulations must be kept to a bare minimum. At the same time little or nothing is known up-front about the true behavior of the system.

Mentioning these assumptions during conference presentations usually results in one of two possible questions or critiques (depending on the audience). A first critique is that these three assumptions limit the space of possible applications to such a degree that they exclude any interesting problems. The scope is thus too narrow. The other critique is that these assumptions are very vague and the problem statement still too general and ambitious. The scope is thus too broad.

In each case counterexamples can be given and this thesis describes many applications that are both interesting and specific enough to tackle and learn from. An excellent motivation and review is given in [9]. The wide range of researchers that have downloaded and made use of the software that resulted from this thesis testifies to this as well.

1.2 Research Challenge

1.2.1 Main challenge

The core challenge of this thesis is illustrated in figure 1.2 and is concerned with *how to efficiently generate an accurate global surrogate model of a computationally expensive simulation code within the requirements specified by the domain expert or engineer.* This task is subject to three constraints, namely that the computational cost and overhead should be minimized, while the surrogate model accuracy should be maximized. Solving this problem involves two main sub-challenges: (1) the generality-specificity trade-off and (2) the accessibility challenge.

1.2.2 Sub-challenge I: generality versus specificity

In its purest form, surrogate modeling treats the computationally expensive simulation code as a data generating black-box, only taking into account its input-output behavior. Thus surrogate modeling is sometimes referred to as behavioral modeling. The advantage of this is that, since no explicit domain specific dependencies are introduced, it becomes possible to leverage the extensive set of techniques and algorithms that have

[1] As an aside, even if the simulation model is deterministic, its parameter values may still be uncertain so risk or uncertainty analysis is needed and the classical notions become relevant again [7, 8]. However, this falls outside the scope of this thesis.

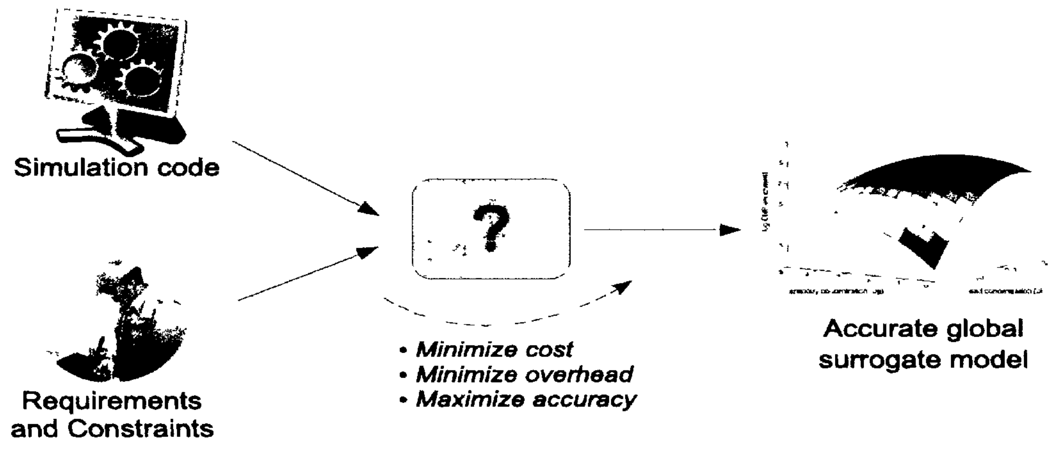


Figure 1.2: Surrogate modeling research challenge

already been developed for regression, approximation, and interpolation. In addition, any new algorithms or tools that are developed have the advantage of being readily portable across different problems and fields.

However, a first problem with this is that there is a very large set of design choices that must be overcome in order to solve the surrogate model generation problem: how must data be collected (design of experiments, adaptive sampling), how should simulations be run (scheduling, distributed computing), what model type and complexity should be used (model selection, hyperparameter optimization), what input variables are most important (feature selection), etc. If little is known about the true response behavior (as stated in section 1.1.2), making informed decisions as to which algorithms should be used is far from trivial given the wide range of options.

A second, more fundamental problem can be found in the well-known aphorism coined by Geffray Mynshul in 1612: *"A jack of all trades is a master of none"*[2]. What you gain in portability and generality, you loose in accuracy and performance for a particular problem. This is the fundamental generality-specificity trade-off which lies at the heart of every machine learning problem and must be tackled somehow.

1.2.3 Sub-challenge II: accessibility

To add to the difficulty, there is a complex dependency web between the different options and sub-problems that make up the surrogate modeling process. Dealing with these dependencies and assumptions is non-trivial, particularly for a domain expert for whom the surrogate model is just an intermediate step towards solving a larger, more important problem. Few domain experts will be experts in the intricacies of efficient sampling and modeling strategies. Their primary concern is obtaining an accurate re-

[2]Actually the full version is quoted as *"Jack of all trades, master of none, though ofttimes better than master of one"*. Food for thought.

placement metamodel for their problem as fast as possible and with minimal overhead. Model (type) selection, model parameter optimization, sampling strategy, etc. are of lesser or no interest to them.

What usually happens in practice is that choices are made in a rather ad hoc manner, driven by (understandably) pragmatic decisions based on the availability of off-the-shelf software, existing experience and common practice within the relevant field, available time and computing resources, etc. [4]. The result is that many (potentially superior) methods go untried with unnecessary waste of computing resources as a result. In addition, the methods that are applied, often are done so in a one-shot, or trial and error based approach [9]. The work flow typically reduces to:

1. assess available computing resources
2. perform simulations until the computing budget is exhausted
3. manually fit one ore more models on the collected data
4. perform some statistical analysis to select a suitable model

A way to improve this one-shot, manual usage of surrogate modeling methods while keeping the modeling process tractable and minimizing the barrier of entry for the domain expert marks the second sub-challenge of this thesis.

1.3 Thesis goal and contributions

A huge amount of comparative studies have already been conducted that try to extract general rules and guidelines with regard to modeling and sampling strategy choices in order to come to an improved work flow [2, 3, 10–21]. However, such studies have very limited generalization power and should always be interpreted carefully (chapter 7 will elaborate on this topic). Domain experts can benefit more from a more holistic, adaptive, and generative approach to surrogate modeling. Given that data is expensive and its use must be optimized, a close integration of sampling, modeling and sample evaluation strategies is paramount. A good discussion of these aspects is given in [5,9] and ideas in this direction have been explored in [22–25].

The motivation for a more adaptive, sequential approach to the surrogate modeling problem also comes from the underlying design process itself. To paraphrase [9]:

> *During the design processes, the design information increases exponentially along the design time line. At different points along the design time line the design requirements and designer knowledge' change. At the beginning period the design efficiency is much emphasized while as design goes on more and more focus is put on the design effectiveness. A designer in the early stages of conceptual design knows little about the problem or*

the design space and does not necessarily know which type of DOE or which metamodeling methods will be most effective and efficient for that particular problem. As the design evolves and more information becomes available, it may be possible to determine which methods are appropriate for that particular problem.

From the viewpoint of surrogate modeling, this shift of design requirements corresponds to the need for an adaptive metamodel strategy with sequential design of experiments.

The goal and contribution of this thesis is to develop such a strategy that goes beyond comparisons on individual sub-problems and aims for seamless, adaptive integration. In addition, given the pace of development, and given the wide range of available techniques, there is a need to easily switch between different algorithms. Be it because the problem or domain expert requires it, or simply to facilitate benchmarking between methods. Furthermore, a problem facing domain experts is that advanced sampling or modeling techniques are described in literature but applying or benchmarking them is often difficult since no implementation is made available. Thus any conceptual framework linking the different sub-problems of surrogate modeling should be accompanied with a software implementation in order to ensure that algorithms can easily be tested, compared, and used by domain experts.

Thus, in summary, the contribution of this thesis is a systematic, yet flexible framework that replaces the "?" in figure 1.2, and that

- minimizes the number of simulations in order to come to an accurate model
- implements novel model management and sample selection strategies, providing automation where possible
- can integrate with the wide range of modeling techniques already available and can be extended to incorporate new methods
- links together the different steps of the modeling process (data collection, model construction, etc.) in a single robust algorithm
- allows a domain expert to quickly apply, compare, and test different methods without requiring a steep learning or installation curve
- easily integrates with other tools of the wider design and engineering pipeline

Hence the subject and focus of this dissertation will be on the meta-level. No special attention will be given to any particular fitting technique or area of application. Rather the thesis will concentrate on the surrogate modeling *process*.

1.4 Thesis organization

This dissertation consists of 11 chapters and two appendices. The first three chapters cover the motivation for this work and fix the canvas within which research took place. This involves an extensive discussion of the motivations and sub-problems of the surrogate modeling problem and how they all fit together.

This is followed by chapter 4 which is an important chapter since it explains the design and implementation of the software framework that forms the core of the tests and algorithms discussed in this dissertation. This is then followed by a first real world case study in chapter 5.

Chapters 6 to 9 build upon the preceding chapters by discussing various extensions and improvements to the classical surrogate modeling algorithms in the distributed, multi-objective, and model selection domains respectively. Chapter 10 then presents a number of illustrative applications, followed by the conclusion and pointers to future work in chapter 11. The thesis is completed by two appendices, one on the problem of generating efficient and accurate circuit models for transistors (appendix A), and one (more informative) appendix providing some background information on the SUMO Toolbox development infrastructure and process (appendix B).

1.5 List of publications

1.5.1 Journals

- **Evolutionary Knowledge-Based Technique for Modeling fo Nonlinear Microwave Devices**
  D. Gorissen, L. Zhang, Q. J. Zhang, T. Dhaene
  Submitted to IEEE Transactions on Microwave Theory and Techniques

- **A Study of Space-Filling Sequential Design Strategies for Adaptive Surrogate Modelling**
  K. Crombecq, I. Couckuyt, D. Gorissen, T. Dhaene
  Submitted to Computers and Structures

- **A Novel Hybrid Sequential Design Strategy for Global Surrogate Modelling of Computer Experiments**
  K. Crombecq, D. Gorissen, D. Deschrijver, T. Dhaene
  Submitted to SIAM Journal of Scientific Computing

- **Blind kriging: Implementation and performance analysis**
  I. Couckuyt, D. Gorissen, T. Dhaene
  Submitted to Journal of the AIAA

- **Automatic surrogate model type selection during the optimization of expensive black-box problems**
  I. Couckuyt, D. Gorissen, T. Dhaene
  Submitted to the Journal of the AIAA, Technical Note

- **On the Use of Machine Learning Techniques for Electronic Design Automation**
  D. Gorissen, K. Crombecq, T. Dhaene, D. Deschrijver
  To be submitted

- **Adaptive Response Modeling with Global Surrogates**
  D. Gorissen, T. Dhaene, E. Laermans
  Submitted to Advances in Engineering Software

- **Estimating Response Nonlinearity in Regression**
  D. Gorissen, I. Couckuyt, T. Dhaene, P. Demeester
  Submitted to Journal of Machine Learning Research

- **Evolutionary Model Type Selection for Global Surrogate Modeling**
  D. Gorissen, F. De Turck, T. Dhaene,
  Journal of Machine Learning Research,
  Vol. 10, No. 1, pp. 2039-2078, September 2009.

- **Multiobjective Surrogate Modeling, Dealing with the 5-Percent problem**
  D. Gorissen, I. Couckuyt, E. Laermans, T. Dhaene
  Springer - Engineering with Computers, 18 pages, *online first*
  http://www.springerlink.com/content/24104526223221u3/

- **Sequential Modeling of a Low Noise Amplifier with Neural Networks and Active Learning**
  D. Gorissen, L. De Tommasi, K. Crombecq, T. Dhaene
  Springer - Neural Computing & Applications,
  Vol. 18, Nr. 5, pp. 485-494, June 2009.

1.5.2 Conferences

- **A Novel Sequential Design Strategy for Global Surrogate Modeling**
  K. Crombecq, L. De Tommasi, D. Gorissen and T. Dhaene
  Proceedings of the 41th Conference on Winter Simulation,
  Austin, Texas, USA

- **Pareto-Based Multi-output Metamodeling with Active Learning**
  D. Gorissen, I. Couckuyt, E. Laermans, T. Dhaene,

INNS Best paper award,
Proceedings of the 11th International Conference on Engineering Applications of Neural Networks (EANN 2009), London (UK),
Springer - Communications in Computer and Information Science,
Vol. CCIS 43, pp. 389–400, August 2009.

- **Space-Filling Sequential Design Strategies for Adaptive Surrogate Modelling**
  K. Crombecq, I. Couckuyt, D. Gorissen, T. Dhaene,
  First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering (CSC 2009), Funchal (Madeira, Portugal),
  Civil-Comp Press, Stirlingshire, United Kingdom,
  paper 50, 20 pages, September 2009.

- **Automated Response Surface Model Generation with Sequential Design**
  I. Couckuyt, K. Crombecq, D. Gorissen, T. Dhaene,
  First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering (CSC 2009), Funchal (Madeira, Portugal),
  paper 52, 15 pages, September 2009.

- **Pareto-based multi-output model type selection**
  D. Gorissen, I. Couckuyt, K. Crombecq, T. Dhaene,
  Proceedings of the 4th International Conference on Hybrid Artificial Intelligence (HAIS 2009), Salamanca, Spain
  Springer - Lecture Notes in Artificial Intelligence, Vol. LNCS 5572,
  pp. 442-449, June 2009.

- **Grid-Enabled Adaptive Metamodeling and Active Learning for Computer Based Design**
  D. Gorissen
  Graduate Student Symposium, Proceedings of The 22nd Canadian Conference on Artificial Intelligence (AI 2009), Kelowna, Canada
  Springer - Lecture Notes in Artificial Intelligence, Vol. LNCS 5549,
  pp. 266-269, May 2009.

- ***Automatic Calibration of Semi-Distributed Conceptual Rainfall–Runoff Model Using MOSCEM Algorithm***
  H. Rouhani, D. Gorissen, I. Couckuyt, J. Feyen,
  Proceedings of the 8th International Congress on Civil Engineering, Shiraz, Iran
  Volume 3 : Water resources and environmental engineering
  pp. 24-32, May 2009

- **Evolutionary Regression Modeling with Active Learning: An Application to Rainfall Runoff Modeling**

I. Couckuyt, D. Gorissen, H. Rouhani, E. Laermans, T. Dhaene,
Proceedings of the International Conference on Adaptive and Natural Computing Algorithms (ICANNGA 2009), Kuopio (Finland),
Springer - Lecture Notes in Computer Science, Vol. LNCS 5495,
pp. 548–558, May 2009.

- **A novel hybrid active learning strategy for nonlinear regression**
  K. Crombecq, I. Couckuyt, E. Laermans, T. Dhaene,
  Beneleam 09, Tilburg (The Netherlands),
  pp. 109-110, May 2009.

- **Surrogate Modeling of Low Noise Amplifiers based on Transistor Level Simulations**
  L. De Tommasi, D. Gorissen, J. Croon, T. Dhaene,
  The 7th International Conference on Scientific Computing in Electrical Engineering (SCEE 2008), Helsinki, Finland
  September 28 - October 3, 2008.

- **Compact modeling of RF circuit blocks via Kriging surrogates**
  D. Gorissen, L. De Tommasi, J. Croon, T. Dhaene,
  3rd Microwave and Radar Week (MIKON 2008), Wroclaw, Poland
  pp. 688-691, May 19-23, 2008.

- **Automatic Model Type Selection with Heterogeneous Evolution: An application to RF circuit block modeling**
  D. Gorissen, L. De Tommasi, J. Croon, T. Dhaene,
  IEEE World Congress on Computational Intelligence (WCCI 2008), Hong Kong (China),
  pp. 989-996, June 2008.

- **Improved Rainfall-Runoff Modeling Combining a Semi-Distributed Model with Artificial Neural Networks**
  H. Rouhani, D. Gorissen, P. Willems, J. Feyen,
  The 4th International SWAT Conference, Delft, The Netherlands,
  July 2-6, 2007.

- **Adaptive Distributed Metamodeling**
  D. Gorissen, K. Crombecq, W. Hendrickx, T. Dhaene,
  Revised Selected and Invited Papers,
  Post-conference proceedings,
  7th International Meeting on High Performance Computing for Computational Science (VECPAR 2006), Rio de Janeiro (Brazil),
  Springer - Lecture Notes in Computer Science, Vol. LNCS 4395,
  pp. 579-588, April 2007.

- **Adaptive Global Metamodeling with Neural Networks**
  D. Gorissen, W. Hendrickx, T. Dhaene,
  15th European Symposium on Artificial Neural Networks (ESANN 2007), Bruges (Belgium),
  pp. 187-192, April 2007.

- **Adaptive Global Surrogate Modeling**
  D. Gorissen, W. Hendrickx, K. Crombecq, W. van Aarle, T. Dhaene,
  SIAM Conference on Computational Science and Engineering (CSE07), Costa Mesa (CA),
  pp. 160, February 2007. *Poster Session*

- **Benchmarking Adaptive Multivariate Surrogate Modeling Techniques**
  D. Gorissen, W. Hendrickx, T. Dhaene,
  SIAM Conference on Computational Science and Engineering (CSE07), Costa Mesa (CA),
  pp. 110, February 2007. *Abstract*

- **Grid Enabled Sequential Design and Adaptive Metamodeling**
  W. Hendrickx, D. Gorissen, T. Dhaene,
  2006 Winter Simulation Conference (WSC '06), Monterey (CA),
  pp. 872-881, December 2006.

- **Grid Enabled Metamodeling**
  D. Gorissen, K. Crombecq, W. Hendrickx, T. Dhaene,
  7th International Meeting on High Performance Computing for Computational Science (VECPAR 2006), Rio de Janeiro (Brazil),
  9 pages (CD ROM), July 2006.

- **Integrating Gridcomputing and Metamodeling**
  D. Gorissen, W. Hendrickx, K. Crombecq, T. Dhaene,
  6th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2006), Singapore (Singapore),
  pp. 185-192, May 2006.

- **Integrating Grid Information Services with JNDI**
  D. Gorissen, P. Wendykier, D. Kurzyniec, V. Sunderam,
  15th International Heterogeneous Computing Workshop (HCW 2006), Rhodes Island (Greece),
  April 2006.

- **H2O Metacomputing - Jini Lookup and Discovery**
  D. Gorissen, G. Stuer, K. Vanmechelen, J. Broeckhove,
  International Conference on Computational Science 2005 (ICCS 2005), Atlanta

(USA),
Springer - Lecture Notes in Computer Science, Vol. LNCS 3515,
pp. 1072-1079, June 2005.

1.5.3 Book chapters

- **Automatic Approximation of Expensive Functions with Active Learning**
  D. Gorissen, K. Crombecq, I. Couckuyt, T. Dhaene,
  Published in: ***Foundations of Computational Intelligence***
  Volume 1: Learning and Approximation: Theoretical Foundations and Applications, Part I: Function Approximation
  Edited by A-E. Hassanien, A. Abraham, A.V. Vasilakos, and W. Pedrycz,
  Springer, series: Studies in Computational Intelligence, Vol. 201,
  ISBN: 978-3-642-01081-1
  Springer Verlag, Germany,
  Chapter 2, pp. 35-62, May 2009.

- **Grid enabled surrogate modeling**
  D. Gorissen, T. Dhaene, P. Demeester, J. Broeckhove,
  Published in: ***Handbook of Research on Grid Technologies and Utility Computing: Concepts for Managing Large-Scale Application***
  Edited by E. Udoh and F. Wang,
  ISBN 978-1-60566-184-1,
  Information Science Reference,
  Chapter 25, pp. 249-258, May 2009.

1.5.4 Technical reports

- **Automatic Approximation of Expensive Functions with Active Learning**
  D. Gorissen, K. Crombecq, I. Couckuyt, T. Dhaene
  UA Technical report TR-10-08,
  22 pages, November 2008.

- **Multiobjective global surrogate modeling, solving the 5 percent problem**
  D. Gorissen, I. Couckuyt, T. Dhaene
  UA Technical report TR-08-08,
  32 pages, September 2008.

- **Introduction to Surrogate Modeling of Narrowband Weakly Nonlinear Low Noise Amplifiers**
  L. De Tommasi, D. Gorissen, K. Crombecq, T. Dhaene
  NXP Technical Report, NXP-R-TN 2008/00293,
  56 pages, August 2008.

- **Towards an Adaptive and Flexible Metamodeling Toolbox**
  D. Gorissen,
  UA Technical report TR-06-14,
  37 pages, November 2006.

- **Applying delegate multi-agent systems in a traffic control system**
  Elise Huard, Dirk Gorissen, Tom Holvoet,
  KULeuven Technical report CW 467,
  25 pages, June 2006.

- **Investigating the Integration of Gridcomputing and Metamodeling**
  D. Gorissen, T. Dhaene, G. Deconinck, B. Dhoedt,
  UA Technical report TR-06-02,
  21 pages, January 2006.

1.5.5 Abstracts - Posters

- **Two level refined direct method for electromagnetic optimization and inverse problems**
  G. Crevecoeur, A. Abdallh, I. Couckuyt, D. Gorissen, L. Depre, T. Dhaene
  Proceedings of Compumag 2009, Florianopolis, Brasil, 2009 (Poster)

- **A comparison of sequential design methods for RF circuit block modeling**
  K. Crombecq, L. De Tommasi, D. Gorissen and T. Dhaene
  Proceedings of the 40th Conference on Winter Simulation, Miami, Florida, December 2008.
  pp. 2942-2942

- **Surrogate Modeling of Low Noise Amplifiers based on Transistor Level Simulations**
  L. De Tommasi, D. Gorissen, J. Croon and T. Dhaene
  Scientific Computing in Electrical Engineering (SCEE 2008),Sept. 28 - Oct. 3, Helsinki University of Technology, Finland

- **Optimization in surrogate model building for RF circuit blocks.**
  L. De Tommasi, D. Gorissen, J. Croon and T. Dhaene
  The European Consortium For Mathematics In Industry (ECMI 2008), June 30 - July 4, University College London, UK

- **Automatic Regression Modeling with Active Learning**
  D. Gorissen, T.Dhaene, E. Laermans,
  Benelearn 2008, 19-20, Spa, Belgium

- **Automatic Surrogate Model Building for Computer Based Design**
  D. Gorissen, T.Dhaene, P. Demeester,

5th European Conference on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2008), Venice (Italy)

- **Adaptive Surrogate Modeling of Complex Systems**
  T. Dhaene, D. Gorissen, W. Hendrickx,
  22nd European Conference on Operational Research (EURO XXII), Prague (Czech Republic),
  Book of abstracts, pp. 136, July 2007.

- **Integration of Grid Computing and Multivariate Macromodeling**
  D. Gorissen, W. Hendrickx, K. Crombecq, T. Dhaene,
  Workshop on Behavioral modeling and approximatiom, University of Antwerp, Belgium,
  21 August 2006.

- **Alternative Approaches to Grids and Metacomputing**
  Tutorial, G. Stuer, D. Gorissen,P. Hellinckx,
  ICCS 2005, Emory University, Atlanta,
  May 2005.

- **H2O Metacomputing - Jini Lookup and Discovery**
  8th Jini Community Meeting, The Brewry, London,
  7-8 December 2004.

1.6 Conclusion

Due to the computational complexity of current simulation codes, the use of global surrogate modeling techniques (adaptive sampling, adaptive modeling) has become popular among scientists and engineers alike. However, considerable problems and choices need to be overcome in order to apply surrogate modeling methods in an efficient and user friendly manner. The goal of this work is to go beyond the traditional one-shot (or even ad hoc) application of surrogate methods by integrating the many advanced solutions to the different sub problems of global surrogate modeling in a unified, pluggable, adaptive framework (both conceptually and literally, i.e., in software). This will go one step further than existing efforts in this area [5, 9]. This framework will facilitate the transfer of surrogate modeling expertise to domain experts and will be useful in any domain where a cheap, accurate approximation is needed for an expensive reference model or data source. To achieve this, the thesis will build on insights from distributed systems, machine learning and multivariate statistics.

2 Surrogate Modeling

On two occasions I have been asked,—"Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?" . . . I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question. "
— Charles Babbage

2.1 Introduction

Having laid out the structure, scope, and motivation of this thesis in the previous chapter we now turn to surrogate modeling itself. However, this is still quite a broad field with many subtopics. Thus, the purpose of this chapter is to provide some historical and conceptual context with regard to modeling and the use of surrogate models in engineering design. This should clarify precisely which aspects of surrogate models are of interest in this work. The chapter starts with some background and history on (computer) modeling, followed by the practical problems involved, and potential solutions through approximation. This brings us to the topic of surrogate modeling which is defined and illustrated with applications from different domains.

2.2 Modeling

Before we consider surrogate modeling, it is useful to first take a step back and consider the motivation for modeling in general.

2.2.1 Background and definition

Since the dawn of civilization, humans have used abstractions of the real world in order to be able to reason about it and the phenomena that occur within it[1]. Actually, anthropologists think that the ability to build abstract models is the most important feature which gave homo sapiens a competitive edge over less developed human races like homo neandertalensis [27]. Or, to quote eminent computer scientist Tony Hoare, "*In the development of the understanding of complex phenomena, the most powerful tool available to the human intellect is abstraction.*".

The very first 'models' were numbers and the writing of numbers (e.g., as marks on bones) and date back to about 30,000BC. With the development of Astronomy and Architecture around 4,000BC, models slowly became more complex and became well used. It is well known that by 2,000 BC at least three cultures (Babylon, Egypt, India) had a decent knowledge of mathematics and used mathematical models to improve their every-day life. Most mathematics was used in an algorithmic way, designed for solving specific problems [27].

From then on the complexity of models (and their application) further continued to increase through the Hellenic and Roman Ages (Thales of Miletus, Aristotle, Euclid, Ptolemey) on to the Middle Ages (Abu Abd-Allah ibnMusa Al-Hwarizmi, Fibbonacci, Vieta) and modern times (Newton, Russel, Einstein). While the emphasis mostly was on mathematical models, other types of models were (are) used as well. Examples include: Visual Models (for example the Anatomy models developed by Vesalius in the early 16th century), structural models (e.g., scale model of an aircraft to test in a wind tunnel), and the more modern biologically inspired models such as Artificial Neural Networks (originally proposed by Warren McCulloch and Walter Pitts in 1943).

In his "A History of Economic Theory", economist J. Niehans claims that in 1894 the "*era had began in which scientists interpreted their activity as model building*" [28, 29]. In this very year, H. Hertz published his famous book "Principles of Mechanics". Therein the famous physicist writes [28]:

> *We make for ourselves internal images or symbols of the external objects, and we make them in such a way that the consequences of the images that are necessary in thought are always images of the consequences of the depicted objects that are necessary in nature . . . Once we have succeeded in deriving from accumulated previous experience images with the required property, we can quickly develop from them, as if from models, the consequences that in the external world will occur only over an extended period or as a result of our own intervention.*

By the end of the 19th century, model building began to dominate the (theoretical) activity in the field of physics: J.C. Maxwell used hydrodynamic analog models to

[1] An accessible overview of the history of abstractions and their use in mathematics is available in [26].

derive the well known equations of electromagnetism and W. Thompson, later Lord Kelvin, stated that he could not understand a phenomenon until he had succeeded in constructing a (mechanical) model of the system under consideration[2] [29].

The word model itself comes from the Latin word *modellus*, a diminutive form of *modulus*, the word for measure or standard. The old Italian derivation *modello* referred to the mould for producing things. In the sixteenth century the word was assimilated in French (*modele*), taking its meaning as a small representation of some object, spreading into other languages like English and German [27]. The Merriam-Webster dictionary defines a model as

> *...a system of postulates, data, and inferences presented as a mathematical description of an entity or state of affairs; also : a computer simulation based on such a system <climate models>*

For a more philosophical treatment of the term model and its use in science the reader is referred to the excellent discussion in [30].

2.2.2 Motivation

What type of problems does modeling attempt to solve, when should one look toward (approximation) models? In general the motivation stems from the desire to understand, explain, reproduce, or predict a particular real world phenomenon (e.g., what are the conditions for it to occur, how does it behave over time, etc.).

According to [27],

> *A model is a simplified version of something that is real.*

A model is a way to represent and structure knowledge about the real world and forms the basis of being able to reason about the world. Since a model is by definition a simplification of the real world, every model construction process incurs some amount of uncertainty that impacts its usefulness. The most crucial part of every modeling process is being aware of this uncertainty and how it limits the application of the model. A classic quote in this respect is the oft quoted dictum by renowned statistician George Box:

> *All models are wrong but some are useful*

A very accessible overview of modeling and its pitfalls is given in [31]. In general models can vary in their level of formality, explicitness, richness in detail, relevance, and ability to generalize. Once constructed, models are used as a basis for:

[2]This quote is attributed to Kelvin by [29] but I did not succeed in finding the original quote in the Baltimore Lectures themselves (where the quote is said to be from). However, the lectures themselves are an interesting read, there Kelvin attempts to formulate a physical model for the aether, that nebulous medium that was seen to be necessary to explain electromagnetic radiation. The lectures are also an important milestone, marking the end of the mechanistic, Newtonian view of science. A view soon to be overturned by Einstein, Planck and others.

- Prediction (e.g., avalanche prediction)
- Interpolation (e.g., obtaining values for missing measurements)
- Extrapolation (e.g., climate extrapolation, will the world be sustainable in the near future)
- Decision making (e.g., classifying tumors based on malignancy)
- Communication (e.g., visual models, 3D structure of DNA)
- Dimensionality Reduction (e.g., clustering of microarray data).
- De-noising (e.g., face recognition)

Different paradigms exist for describing the modeling process and each scientific discipline has their own ideas about specific types of modelling. Some general theory about modeling can be found in philosophy of science, systems theory, and new fields like knowledge visualization. A good overview is given in [29].

2.3 Computer modeling and simulation

For much of human history modeling was restricted to the pencil and paper or thought experiment variety. Based on painstaking observation and careful ingenious analysis, models were manually constructed for many natural phenomena. The examples from astronomy (Kepler, Hubble, Newton, and others) being perhaps the most well known. One can only marvel at the insight, discipline, and meticulous experimentation of scientists in those times (e.g., over eight years, Mendel grew an estimated 28,000 pea plants). However, much of this changed with the rise of mechanical devices and later, computers. This allowed a scientist to perform highly controlled virtual experiments on computer hardware through the use of simulations. This marked a complete paradigm shift, or as as F. Rohrlich and others [32] have emphasized, computer simulations provided "*a qualitatively new and different methodology . . . that . . . lies somewhere intermediate between traditional theoretical physical science and its empirical methods of experimentation and observation*".

The use of simulation[3] drastically increased the breadth and depth at which phenomena could be studied and predicted and allowed the solution of many problems where pure analytical methods had failed.

2.3.1 Historical background

The history of computer simulation dates back to World War II where scientists Jon Von Neumann and Stanislaw Ulam were faced with the problem of neutron behavior as part

[3]The word simulation can have many meanings and interpretations. A good overview is given in [33].

of the Manhattan project [34]. With the remarkable success of the techniques on the neutron problem, it soon became popular and found many applications in business and industry. However, even then simulation was still a complex, time consuming process that required a great deal of ingenuity in order to setup the necessary hardware and produce stable solutions. The analog computers that were typically used to simulate differential equations were tricky to handle and needed to be manually interconnected and configured.

Commercially designed computers (both analog and digital) appeared in the late 1940s and early 1950s but computer simulation was still not cost effective as a tool. Simulation took too long to get results, needed too many skilled people, and as a result cost a considerable amount in both personnel and computer time. And most disheartening, results were often ambiguous.

Things improved in the 1960s when in December 1961 Geoffrey Gorden from IBM presented his paper at the fall Joint Computer Conference on a General Purpose Systems Simulator (GPSS) [35]. GPSS came into existence rapidly, with virtually no planning, and surprisingly little effort. It came rapidly because it filled an urgent need that left little time for exploring alternatives. It was, however, very successful and marked the start of a simulation community through the IBM user's group conference SHARE. Meanwhile other companies were developing similar tools and languages (e.g., SIMSCRIPT developed by Rand Corporation). The creation of Simula by norwegian scientists Kristen Nygaard and Ole-Johan Dahl [36] also stems from this time. Simula was a special purpose programming language for simulating discrete event systems and is generally regarded as the first object oriented language, strongly influencing later languages like SmallTalk, C++, and Java. However, in general there were little efforts to coordinate and compare the different approaches to tackling simulation problems.

These limitations were addressed through a series of workshops and conferences, the first being a workshop on Simulation Languages at Stanford University in March of 1964. Consolidation efforts culminated in the first Conference on Application of Simulation using the GPSS in November 1967. The popularity of this conference grew each year and in its fifth edition in 1971 it was renamed to The Winter Simulation Conference which continues to this day. The increasing number of conferences and periodicals throughout the 1970s marked the maturity of the field and the establishment of a common forum and conceptual framework for simulation-based research.

In the 1980s hardware and software prices continued to drop while reliability, performance, and accessibility increased significantly. This resulted in an exponential explosion of simulation applications and platforms, ultimately leading to the ubiquitous use of simulation we see today. A nice illustrative quote of the importance of simulation is given in [29].

Major parts of current research in the natural and social sciences can no longer be imagined without simulations, especially those implemented on a computer, being a most effective methodological tool. Natural scientists *simulate the formation and development of stars and whole galaxies, the detailed dynamics of violent high-energy nuclear reactions as well as aspects of the intricate process of the evolution of life, while their colleagues in the social science departments simulate the outbreak of wars, the progression of an economy and decision procedures in an organization – to mention only a few. Recently, computer simulations even proved useful in moral philosophy. In fact, there is almost no academic discipline without at least a little use for simulations.*

A fascinating and accessible account of the meaning, importance and use of simulation can be found in [29].

2.3.2 Computer modeling process

At the heart of computer-based simulation (versus experimental simulation) lies the computer modeling process. This brings us to the simplified model development process as introduced by Sargent [37] and depicted in figure 2.1.

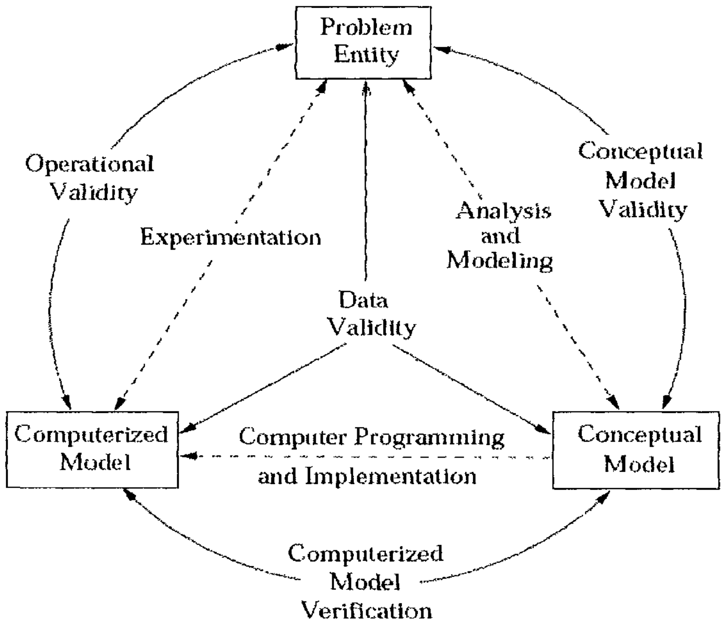


Figure 2.1: Simplified version of the modeling process [37]

From figure 2.1, the problem entity is the system or real world phenomena to be modeled. In order to come to a computerized model of the problem, the problem entity must first be captured in a mathematical/logical/verbal/... conceptual model through an analysis and modeling phase. This is carried out by the domain experts of the problem in question. The computerized model is then the conceptual model implemented through a computer programming and implementation phase. Once a computerized model is available, it can be used for inference about the problem entity in the experimentation phase.

Checking that the theories and assumptions underlying the conceptual model are correct and 'reasonable' for the intended purpose of the model is represented by the conceptual model validity arc. Likewise, the computerized model verification arc is defined as ensuring that the computer programming and implementation of the conceptual model is correct. Determining that the model's output behavior has sufficient accuracy over the domain of the model's intended applicability is the purpose of operational validity. Finally, data validity is defined as ensuring that the data necessary for model building, model evaluation and testing, and conducting the model experiments to solve the problem are adequate and correct.

For the purpose of this thesis we will mainly focus on the computerized model, the problems it introduces, and how we can further abstract it to alleviate these problems.

2.4 Limitations of computer models

We have already touched on the importance of understanding the assumptions, uncertainty, and application domain associated with a model. The same of course applies to computer-based simulation, the popular saying being *"Garbage In, Garbage Out"* (GIGO). Or, as Lee [38] put it: *"Bigger models simply permit for bigger mistakes"*. Karplus also elaborates on this topic in [39, 40].

However, for this section we are more interested in the practical limitations of computer-based simulation, rather than those related to the problem domain and interpretation.

2.4.1 Computational cost

The most fundamental problem with computer-based simulations is that they are computationally expensive to run. One simulation may take many minutes, hours, days, or even weeks [3, 41–43]. If a simulation need only be run a couple of times, e.g., the system under study and the simulation parameters are already well understood, this may be acceptable. However usually a scientist may wish to repeat a simulation many times using different parameters or starting from different initial conditions. This quickly becomes intractable. Perhaps those most affected by this problem are applications from engineering design.

There exists a huge range of design problems from electro-magnetism, aerodynamics, automotive, and related fields where engineers routinely have to perform computer simulations in order to design, build, and test, new devices or mechanical parts. The problem is amplified by the fact that the simulations are usually parametrized. The design variables of the problem determine a parameter space that needs to be mapped or searched in order to find efficient new designs that meet the application specifications. This typically involves such data intensive tasks as optimization, design space exploration, sensitivity analysis, feature selection, etc. As the cost of a single simulation grows this quickly becomes prohibitively expensive. Note, however, that the cost of one simulation need not be on the order of hours in order to impact the design process [44].

Nearly thirty years ago, Goodman and Spence [45] found that response delays as little as 1.5 seconds in the design software can increase task completion time by approximately 50%. They examined the effect of system response delay on the time to complete an artificial task that was created to mimic design activity. The task was the graphical adjustment of five parameters to change the shape of a function (presented graphically) so that it passed between forbidden regions in the $x, f(x)$ plane [46].

In [47], Simpson et. al. strive to determine the efficacy of metamodel-driven visualization for graphical design and optimization. They discern two categories in their investigations: (1) assessing the benefit of having a rapid response to user requests for performance as a function of design parameters, and (2) assessing the cost of lost accuracy due to the use of approximations or metamodels. In particular they discuss results from a study involving a wing design problem where the response delay and number of variables can be changed in a controlled fashion to measure the impact this has on the design efficiency and solution strategy. They found that

> *Response delay and problem size both had a significant effect on design effectiveness: the 1.5 second delay increased average error by 150% compared to the no delay case, and the average error more than doubled each time the problem size increased. Problem size also had a significant effect on design efficiency: the average completion times for the 4 and 6 variable problems were more than double the completion time for the 2 variable problem.*

They describe a similar study in [46], involving an I-beam design problem. They observe that

> *Experimental results indicate that, on average, error increased by 280% and completion time increased by 33% when a delay of 1.5s was present, and the perceived workload significantly increased as well.*

They conclude in [47]:

> *Based on our findings, analyses that require more than 1.5 seconds of computation time should be replaced with appropriate metamodels, which have minimal delay to avoid the adverse impacts of response delay in the user interface. Furthermore, limiting the size of the problem will enable users to find better designs in shorter periods of time;*

2.4.2 Large scale systems

A second limitation of high-fidelity computer simulations, related to the computational cost, emerges when simulating large scale systems (e.g., global climate change, electronic devices, complex mechanical machines) or integrating different multi-physics codes. Modeling a complex system like the Earth [48], for example, with accurate simulation models would require a huge number of domain specific simulators to work together coherently. This is not only impossible from a computational point of view, simply getting all the software to interoperate and run is already a non-trivial task.

A classic example is the full-wave simulation of an electronic circuit board. Electromagnetic modeling of the whole board in one run is almost intractable. Instead the board is modeled as a collection of small, compact, accurate models that represent the different functional components (capacitors, transmission lines, resistors) on the board. Examples of such applications can be found in [5,48–50].

2.4.3 Legacy reference model

Finally, a last practical limitation of computer models is that the code may be proprietary and thus prohibitively expensive and inaccessible to scrutiny or modification. Alternatively, the simulation code itself or the platform it runs on may be unmaintained or legacy code, making it arcane to setup and use. On the other hand, the code may be open to modification and improvement but may require a large computer cluster to run in order to produce any useful results. This may pose a barrier if the necessary hardware is not available.

2.5 Approximation approaches

There are different ways to deal with the practical limitations mentioned above. Concerning computational cost the easiest solution is to simply throw more hardware at the problem and let Moore's law do the rest. However, this is not always a solution. The continuous drive to finer time scales, increasing detail, and provable complexity of core algorithms more than enough offset the advances in computing speed and storage capacity. This work is concerned with a solution based on approximation. The idea is to reduce the overall running time by applying sound approximation methods, resulting in approximation models that can be used inplace of the costly reference model. This brings with it the following advantages:

1. overcoming the computational expense of engineering simulation codes
2. much improved integration of application-independent multidisciplinary codes and simulations
3. allowing for computer/operating system platform independence

4. gaining insight from black-box engineering models
5. validation & verification of the underlying simulation model (see for example [51,52])
6. enabling web-based design and execution

These advantages stem from the simple formulation of an approximation model and its execution speed.

Roughly speaking, three different approaches have been developed to generate approximations: model-driven, data-driven and hybrid approximation. A short overview is given below, an extensive taxonomy can be found in the excellent report by Janssen et. al. [53] or in the book by Keane and Nair [4]. For the more philosophically inclined a good discussion is given in [30].

2.5.1 Model-driven

Model-driven approximation is known as Model Order Reduction (MOR)[4] [56, 57], phenomenological approximation [58], physics based modeling [4], or system theory based metamodeling [53]. Taking a top-down approach, the approximation procedure starts from the original simulator equations and derives approximations using rigorous mathematical techniques [59, 60]. There is a tight relation to the field of numerical linear algebra and most methods are based on Krylov subspace methods (e.g., [61, 62]). Another important aspect is the calculation of dominant eigenvalues and singular values. An example is Proper Orthogonal Decomposition (POD) in computational fluid dynamics (also known as principal components analysis or Karhunen-Loeve in other fields) or spectral decomposition (also known as modal analysis) in structural dynamics [63].

Thus the construction of the model typically involves finding a 'lower order' set of equations that approximate the original set using state-of-the-art algebraic techniques and projection operators. Possible applications lie within domains where the internal dynamics are of primal importance and can be stated explicitly. Examples can be found in electronics, large scale transportation and flow models, and in system-dynamic process models from ecology, economics and demographics. See for example the stock-and-flow models in [64]. Some comparisons between different model-driven methods can be found in [65,66].

[4]Unfortunately the terminology is not always consistent. While the term Model Order Reduction seems to be the most prevailing term at present, the term Reduced Order Modeling (ROM) is also used (e.g., [54]). In both cases they apply to dynamic, time (and frequency) dependent systems. If, besides frequency/time, other parameters are involved in the order reduction (e.g., geometric parameters, conductivity parameters, etc.) the methods may also be referred to as Parametrized MOR (PMOR) [55].

2.5.2 Data-driven

At the other extreme, data-driven (or data-fitting) approximation takes a bottom-up approach [67]. The exact, inner working of the simulation code is not assumed to be known (or even understood), solely the input-output behavior is important (e.g., [68]). A model is constructed based on approximating the response of the simulator to intelligently chosen input configurations. The simulator is usually deterministic and its dynamics, if present, are typically ignored. Due to the black-box approach, data-driven modeling can be applied to almost any domain, be it ecology, economics or physics. The main downside of these methods is that they incorporate no problem specific information and thus lack traceability. Therefore they are often combined or extended to include such information (see the next section).

This approach is also known as behavioral modeling, black-box modeling, or response surface modeling. Good overview references can be found in [2,4,8,44,67,69]. Examples of data-driven approximation methods are plentiful, with low order polynomial regression models being the archetypal example:

- Polynomial/Rational functions
- Radial Basis Function (RBF) models
- Multi Layer Perceptrons
- RBF Neural Networks
- Support Vector Machines
- Regression Trees
- Kriging models
- Gaussian Process models
- Multivariate Adaptive Regression Splines
- Generalized Linear Interactive Modeling
- Generalized Additive Models
- Classification and Regression Trees
- Fuzzy and Neuro-fuzzy methods
- Genetic Programming

2.5.3 Hybrid

Finally, there is a large gray zone where the two overlap: data-driven modeling may include problem specific rules, coarse models, or constraints (e.g., the enforcement of passivity when modeling a passive electronic component or circuit) and model-driven modeling may incorporate simulation data as a further approximation or validation step.

The final models thus combine expert knowledge over the system, that has been abstracted and simplified into analytical expressions or rules, together with (empirical) information about the dynamic system characteristics (e.g., impulse-response functions, transfer functions, etc.) to come to a model where the process knowledge is explicitly represented. These models are useful for extrapolation since the expert knowledge is explicit. The application domain, however, is more restricted since the simplified formulations could fail to capture subtle non-linearities. In any case it is extremely important to explicitly state under what conditions the simplifications are valid.

Hybrid models (also called process-based metamodels) have been successfully used in climate research [70, 71], hyrdrologics [72] and electronics [73]. Good examples are the Knowledge Based Neural Networks (KBNN) developed by Zhang et. al. [74], knowledge based kriging models [75], and the various Space Mapping methods [76–79]: If a simplified approximate simulator is available (referred to as the *coarse model*) as well as the original, high-fidelity simulator (referred to as the *fine model*), a data-driven model can be used to map the former onto the latter. Different ways have been developed to do this, references include [24, 43, 74].

2.5.4 Comparison

Data-driven modeling makes no assumptions (which is both a strength and a weakness). It can be applied to any problem where the process can be described as a data generating black-box. This is useful for systems where the governing equations are not yet fully understood or known, or when the available simulation code is such that it cannot be altered (proprietary or legacy code) or simply not available. In contrast, model-driven modeling has the advantage of being intimately tied to the original partial differential equations (PDEs) and retaining their physics (though performing and validating the simplifications is not trivial). However, often it is the global input-output behavior that is important (e.g., will the total energy in the system be conserved, what is the maximal total force that a structure will hold, etc). In this case, all the approximation intricacies of the different subsystems in the global system no longer play a role and a full model-driven approximation may not be worthwhile.

A nice summary is given by [80]:

> *Some statisticians, operations researchers, and computer scientists prefer the first approach and want to know nothing about the "innards" of the model whose behavior they are attempting to replicate. They may have a purist philosophy of "allowing the data to speak," without "contaminating it" with theoretical assumptions. Or they may simply prefer not having to deal with the complexities of the model's innards: they may wish to turn the problem over to automated software. At the other extreme, some theoretically inclined academicians clearly prefer the second approach because it allows rigorous tying together of phenomena at different levels of detail (as when classical thermodynamics is understood from quantum mechanics). These, then, are the extremes. Most scientists, engineers, and analysts, however, should prefer something in between.*

2.6 Surrogate modeling

Each of the different approximation methods has its place and usefulness. The focus of this thesis is on the data-driven and, to a lesser degree, hybrid variety. We are interested in mimicking the simulator by only taking into account its input-output behavior. Besides the motivation given in the previous subsections, other reasons for their use include [58, 80]:

- to replace or seamlessly integrate one or more legacy reference models that are old, opaque, and difficult to work or interface with
- to link together different multi-physics codes
- to enable optimization
- to allow exploratory analysis, often there is a need to explore the behavior of a model over a large part of its domain
- to reduce the number of variables or perform sensitivity analysis
- to facilitate visualization of the design space, e.g., [81]
- to help construct prototypes and quickly gain insight into the behavior of the system

These types of models are commonly referred to as response surface models, metamodels, behavioral models, and surrogate models.

2.6.1 Terminology

The term Response Surface Methodology (RSM) is perhaps the oldest and is usually associated with the simple low-order polynomials used for modeling experimental data. The methodology has its roots in the seminal paper by Box and Wilson in 1951 [82] that resulted from their collaboration at a chemical company when solving the problem of determining optimal operating conditions for chemical processes. Two decades later the term metamodel, a "model of a model", was coined by Jack Kleijnen [83]. It covers a wider range of techniques (often stochastic in nature) and has its roots in operations research. The term intuitively expresses the extra layer of abstraction incurred (figure 2.2).

The definition given by Davis in [58] is:

> *A metamodel is a relatively small, simple model intended to mimic the behavior of a large complex model, called the object model, that is, to reproduce the object model's input-output relationships*

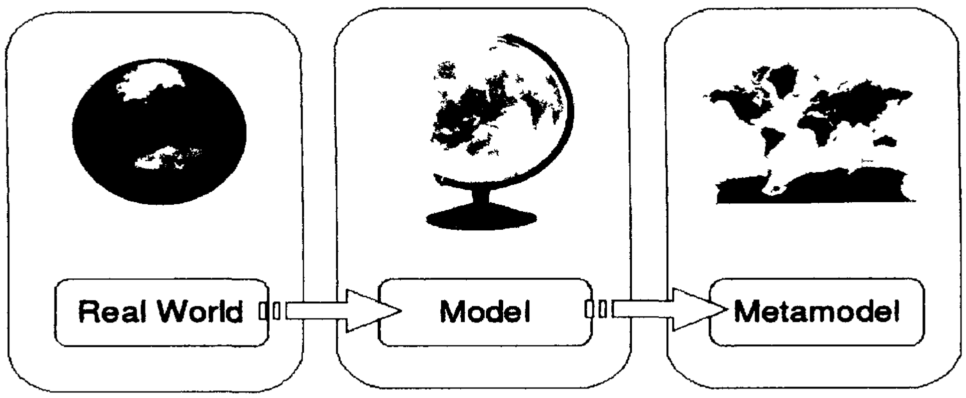


Figure 2.2: Modeling Hierarchy

Another term that is often used is the term surrogate model. The origin of the term is less clear-cut though it also seems to have its roots in operations research where it is cited as far back as in 1959 [84]. It also seems to be used sporadically in medicine as well. While some authors do consider them different [5], in this thesis we consider the terms metamodel and surrogate model to be synonyms.

A typical workflow when using surrogate models for engineering design is as follows (based on [5]):

1. *Model formulation* involves understanding the problem, defining the design objectives, and identifying the problem's input and output parameters; this may include specifying the names and bounds of the variables that will be part of the design, as well as characterizing the responses. At this point, it is also appropriate to determine whether the use of a metamodel is justified, or whether the analysis should be conducted with the original reference model.

2. *Design selection* entails choosing the type of experimental design that will be applied to run the simulator. The true output responses obtained from these runs are used for fitting the surrogate model.

3. *Metamodel fitting* requires specifying the type and functional form of the surrogate model.

4. *Assessment of the surrogate model* involves specifying the performance measures that will be used to characterize the fidelity of a metamodel, as well as choosing an appropriate validation strategy.

5. *Gaining insight* from the surrogate model and its error permits identifying important design variables and their effects on response variables. This is necessary to understand the behavior of the reference model, or redefine the region of interest in the design space.

6. *Using the surrogate* to predict responses at untried inputs and performing optimization runs, trade-off studies, or further exploring the design space.

2.6.2 Global versus local

It is important to make a distinction between two different applications of surrogate models. The first is by far the most popular and involves building small surrogates for use in optimization. This is known as Surrogate Based Optimization (SBO) or Metamodel Assisted Optimization (MAO). With SBO the surrogate model itself is not the main goal but used to approximate the fitness landscape to drive the optimizer. The surrogate models in use are often quite simple (cfr. traditional Response Surface Methodology), with polynomial regression being the most popular among practitioners [85]. This, of course, need not be the case. For example, complex ensemble methods combining different surrogate model types have been used [86, 87], as have been innovative trust region methods [88] and sampling techniques [89, 90]. Excellent overview references of SBO are given by [4, 44, 63, 86]. Well known examples in this category are the proprietary tools developed by LMS/Noesis (Optimus, Virual.Lab)[5] and Vanderplaats R&D (VisualDOC)[6]. From academia, the more prominent projects are Geodise [91] from the University of Southampton (now commercialized as dezineforce), Nimrod/O [92] from Monash University, and the DAKOTA toolkit from Sandia National Labs [93]. A good review of available tools in this category is also given in [2, 44].

In the second case one is not interested in finding the optimal parameter vector but rather in the global behavior of the system. Here the surrogate is tuned to mimic the underlying model as closely as needed over the complete design space. This is the focus of this thesis. Such surrogates are a useful, cheap way to gain insight into the global behavior of the system. Optimization can still occur as a post-processing step but is not the main goal. In addition, they can cope with varying boundary conditions. This enables them to be chained together in a model cascade in order to approximate large scale systems. A good overview of this respect is given in [5]. Even if optimization is the goal, one could argue that a global model is less useful since significant time savings could be achieved if more effort were directed at finding the optimum rather than modeling regions of poor designs. However, this is the logic of purely local models, but they forgo any wider exploration of radical designs [94].

Of course the dichotomy is not strict; ideas and approaches between the two types can, and should, be exchanged, allowing for different hybrids to emerge that borrow ideas from both types. A good example in this respect is the popular Efficient Global Optimization (EGO) approach first described by Jones et. al. in [89] and elaborated by many others (e.g., [90]). In the EGO approach a global model is constructed to

[5]`http://www.lmsintl.com/`
[6]`http://www.vrand.com/`

capture the complete design space and a sampling function is used to refine it in the neighborhood of the optima. In this way the chances of getting trapped in a local optimum are reduced [95] (see also section 4.6).

Concerning global models, one could argue that in order to obtain an accurate global surrogate one still needs to perform numerous simulations, thus still having to deal with the high computational cost of simulations. While naturally the methodology has its limitations, this is not entirely the case since: (1) building a global surrogate is a one-time, up-front investment (assuming the problem stays the same), (2) distributed computing can speed-up the evaluation time and (3) adaptive modeling and adaptive sampling (sequential design) can drastically decrease the required number of data points to produce a good model. However, while intelligent modeling and data collection schemes can extend the application range of global approximations, the infamous curse of dimensionality is inescapable. As the number of dimensions increases the number of points needed to maintain a high global accuracy increases exponentially. At that point one must relax the accuracy requirements, reduce the dimensionality (e.g., through factor screening [96]), or turn towards other approximation methods (cfr. section 2.5).

2.6.3 Forward versus inverse

There is also a distinction between forward surrogate modeling and inverse surrogate modeling and often the motivation for the former is to be able to perform the latter. In classic forward surrogate modeling, the surrogates provide estimates of simulation outputs as a function of design parameters. However, often in the design of a system or product, one has performance targets in mind, and would like to identify system design parameters that would yield the target performance vector [5, 97]. This is illustrated in figure 2.3.

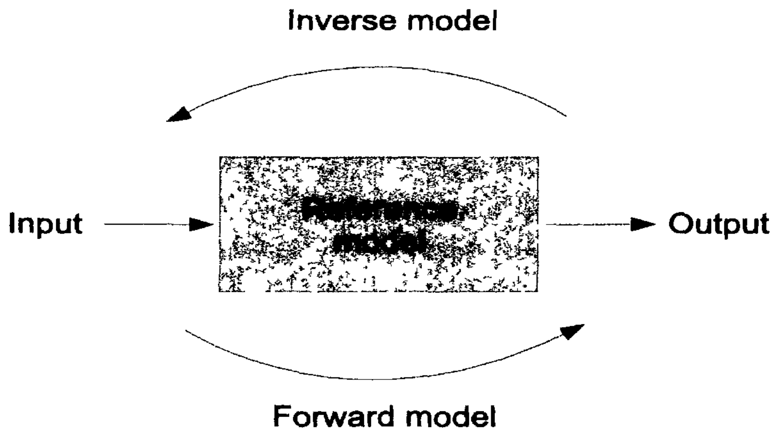


Figure 2.3: Forward versus inverse modeling.

Typically, this is handled iteratively through an optimization search procedure, possibly with robustness constraints [98]. As an alternative, one could map system performance requirements to design parameters via an inverse surrogate model [99]. However, this is a difficult problem since the inverse mapping from function value to design variable values is not uniquely defined.

2.6.4 Applications

At heart, the surrogate modeling problem is a very generic problem. Therefore it comes as no surprise that surrogate models have found applications in many diverse fields where they are used to approximate some complex and/or expensive reference model. To illustrate the diversity of applications some examples are listed below:

- **Economics**: Sensitivity analysis in investment problems [100]
- **Operations research**: modeling buisnes networks [101]
- **Robotics**: evolution of gait patterns of four-legged walking robots [102]
- **Electronics**: mobile antenna design [103]
- **Physics**: study of proton beams [104]
- **Chemistry**: prediction of fibrinogen absorption onto polymer surfaces [105]
- **Automotive**: study the effect of a frontal impact on a vehicle [13]
- **Environmental Science**: studying the vulnerability of ground water to pesticide leaching [106]
- **Biology**: prediction and explanation of biodiversity data [107, 108]
- **Geology**: modeling of (oil, gas, water, ...) reservoirs [109]
- **Meteorology**: studying the effect of emission reduction on ozone concentrations [110]
- **Sociology**: modeling innovation diffusion [111]
- **Medicine**: modeling colon coloration [112]

2.6.5 Words of caution

It is clear that surrogate modeling methods have a very wide application domain. However, as any tool or technique, surrogate modeling has its restrictions and it is important to be well aware of these.

For example, it makes no sense to apply surrogate modeling methods if the simulator is simple and cheap to evaluate. Enough data can easily be generated and there is no need for a second level of abstraction. There is one exception to this rule though. In some cases too much data is available. For example, a dense data set may be generated as part of a production plant measurement process or historical aggregation. In this case surrogate modeling methods can be useful to 'summarize' the data in an efficient, re-usable, analytical model (an example is given in section 10.6).

The very first step in the surrogate modeling process should always be a critical one: Is surrogate modeling the best way to accomplish the necessary goals? Maybe faster machines, parallel computing, more manpower, analytical methods, a faster implementation of the simulator, etc. are more cost effective solutions. The answer to this should depend on a thorough evaluation of:

- the available time, money, manpower, software, expertise for surrogate modeling
- the available expertise about the original simulator
- the need for extrapolation and traceability of the underlying physics
- where and how often the surrogate model will be used
- how often the surrogate model will need to be updated to reflect changes in the simulator

As we shall see in the next chapter, surrogate modeling has its own set of challenges: experimental design, sample selection, model selection, etc. These should also be taken into account. Finally, one should always remember that a surrogate model is only as good as the available data and designer. Surrogate models are still models, making model assessment & selection crucial steps in the design process.

2.7 Conclusion

This chapter reviewed the modeling problem in a broad context, discussing history, motivation, and application. Throughout the sections the topic was narrowed down to what is the core subject of this thesis: forward global surrogate modeling methods based on a (mainly) data-driven approach. The motivation for the use of these methods should be clear as well as how they relate to more model-driven methods. For readers needing more information about the general surrogate modeling domain, good references can be found in [2,4,44,67,69].

3 Adaptive Global Surrogate Modeling

What we observe is not nature itself, but nature exposed to our method of questioning.

– Werner Karl Heisenberg

3.1 Introduction

With the context of the surrogate modeling problem defined we can now take a look at how surrogate models can be constructed. This chapter discusses the different aspects and trade-offs that come into play in order to apply surrogate modeling successfully. Each of the different surrogate modeling sub-problems is treated in detail as well as how they may be integrated into a single algorithm.

3.2 Problem formulation

The mathematical formulation of the problem we are dealing with is as follows:

Approximate an unknown multivariate function $f : \Omega \mapsto \mathbb{C}^q$, defined on some domain $\Omega \subset \mathbb{R}^d$, whose function values $Y = \{f(\mathbf{x}_1), ..., f(\mathbf{x}_k)\} \subset \mathbb{C}^q$ are known at a fixed set of pairwise distinct sample points $X = \{\mathbf{x}_1, ..., \mathbf{x}_k\} \subset \Omega$. Constructing an approximation requires finding a suitable function $\tilde{f}$ from an approximation space S such that $\tilde{f} : \Omega \mapsto \mathbb{C}^q \in S$ and $\tilde{f}$ closely resembles f as measured by some criterion ξ, where ξ consists of three parts:

$$\xi = (\Lambda, \varepsilon, \tau) \tag{3.1}$$

Λ is the model quality estimator with $\Lambda : S \mapsto \mathbb{R}^+$ (lower is better), ε the error (or loss) function, and τ is the target value required by the user. This means that the global surrogate model generation problem (i.e., finding the best approximation $\tilde{f}^* \in S$) for a given set of data points $D = \{(\mathbf{x}_1, f(\mathbf{x}_1)), ..., (\mathbf{x}_k, f(\mathbf{x}_k))\}$ can be formally defined as

$$\tilde{f}^* = \arg\min_{t \in T} \arg\min_{\theta \in \Theta} \Lambda(\varepsilon, \tilde{f}_{t,\theta}, D) \tag{3.2}$$

such that

$$\Lambda(\varepsilon, \tilde{f}^*_{t,\theta}, D) \leqslant \tau \tag{3.3}$$

where $\tilde{f}_{t,\theta}$ is the parametrization θ (from a parameter space Θ) of $\tilde{f}$, and $\tilde{f}_{t,\theta}$ is of model type t (from a set of model types T).

The first minimization over $t \in T$ is the task of selecting a suitable approximation model type, i.e., a rational function, a neural network, a spline, etc. This is the model type selection problem. In practice, one typically considers only a single $t \in T$, though others may be included for comparison. Then given a particular approximation type t, the task is to find the hyperparameter assignment θ that minimizes the model quality measure Λ (e.g., determine the optimal order of a polynomial model). This is the hyperparameter optimization problem, though generally both minimizations are simply referred to as the model selection problem. Many implementations of Λ have been described: the hold-out, bootstrap, cross validation, jack-knife, Akaike's Information Criterion (AIC), etc.

In order to construct $\tilde{f}$ the dataset D needs to be populated. Traditionally the size and distribution of D is chosen up-front. However, since $f(\cdot)$ is expensive to compute it becomes important to avoid unnecessary simulations. However, since the complexity of the response surface is not known up-front, defining an a priori data distribution is difficult. Instead data points (also known as support points, samples, or design sites) must be selected iteratively, at locations where the information gain will be the greatest. Mathematically this means defining a sampling function

$$\Phi(D_j) = X_{j+1} \text{ with } j = 0, .., N \tag{3.4}$$

which proposes a new set of sample points based on the data available so far. This effectively results in a set of nested subsets:

$$D_0 \subset D_1 \subset D_2 \subset ... \subset D_N \tag{3.5}$$

In order to construct D_0, which seeds the sampling function Φ, an initial set of sample locations X_0 is needed. We refer to X_0 as the *initial experimental design* and it is constructed using one of the many algorithms available from the theory of Design and Analysis of Experiments (DoE) (section 3.6).

Since the models need to be updated at each sampling iteration this procedure effectively results in not one, but a sequence of models

$$\tilde{f}^*_{D_0}, \tilde{f}^*_{D_1}, \ldots, \tilde{f}^*_{D_N} \tag{3.6}$$

where $\tilde{f}^*_{D_j}$ is the model trained on dataset D_j and resulting from the minimization in equation (3.2).

The task of Φ is to generate a new set of maximally informative samples based on one or more criteria ϕ. Thus this is again an optimization problem based on the set of models and data available so far:

$$X_{j+1} = \arg \min_{X_l \subset (\mathcal{X} \setminus \bigcup_{i=0}^{j} X_i)} \phi(X_l, \{\tilde{f}^*_{D_0}, \ldots, \tilde{f}^*_{D_j}\}, D_j) \tag{3.7}$$

and subject to the constraint that the number of data points $|X_{j+1}|$ selected each iteration should be minimized while at the same time maximizing the benefit to the model $\tilde{f}$ with respect to the criteria ξ [1].

This process is called adaptive sampling, but is also known as active learning [113], reflective exploration [73], Optimal Experimental Design [114], Sequential Exploratory Experimental Design [9], and sequential design [115]. Adaptive sampling is often applied together with Kriging models. An excellent overview of this work can be found in [8].

3.3 Theoretical remark

As stated above, the objective of surrogate modeling is to generate an approximation surface, based on a limited set of samples, for an unknown function $f(x)$. Considering this objective it is interesting to recall some interesting related theoretical work by Kolmogorov and others.

In 1900 the famous German mathematician David Hilbert gave a memorable lecture at the Second International Congress of Mathematicians in Paris. During his lecture he listed 23 conjectures, hypotheses concerning unsolved problems which he considered the most important outstanding mathematical problems of the 20th century.

[1] A possible critique of the iterative procedure explained here is that it has no foresight, or in optimization terms, it performs a local search. In a way this is true. The described iterative procedure will always optimize the model type and structure in function of the data available at that point in time (ignoring non data-based model selection knowledge for the moment). So the procedure treats every sampling step as potentially the last one, there is no higher level planning or foresight. This is where some authors make the distinction between passive and active sequential design. So in that sense the search is local, and the best model at time $t+1$ may turn out quite different from the one at time t. But as more data arrives the iterative procedure will home in on the correct solution (assuming the model keeps following the data).

His 13th conjecture stated that there exist continuous multivariate functions which cannot be decomposed into a finite superposition of continuous functions of fewer variables [116]. In 1957 the eminent Russian mathematician Vladimir Arnold disproved Hilbert's hypothesis [117], shortly followed by Kolmogorov [118] who proved (with constructive proof) that any continuous function of n dimensions can be completely characterized by a 1-dimensional continuous function. Mathematically, Kolmogorov's original theorem can be stated as follows [116, 119]:

Theorem: *For all $n \geq 2$, and for any continuous real function f of n variables on the domain $[0,1]$, $f : [0,1]^n \rightarrow \mathbb{R}$, there exist $n(2n+1)$ continuous, monotone increasing univariate functions on $[0,1]$, by which f can be reconstructed according to the following equation*

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{q=0}^{2n} \phi_q \left(\sum_{p=1}^{n} \psi_{pq}(\mathbf{x}_p) \right) \tag{3.8}$$

The functions $\psi_{pq}(\mathbf{x}_p)$ are universal for the given dimension n and independent on f. ϕ_q does depend on f and is a continuous, one-dimensional function which totally characterizes $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ (ϕ_q is typically highly non-smooth). Consequently, *"we see that the approximation problem is not so much the dimensionality, but the complexity of the function (high dimensional functions typically have the potential to be more complex)"* [119].

This fundamental approximation theorem is but one of the many intruiging results of research on the fundamental properties of optimization, learning, computing, and approximation algorithms. The work by Goedel, Turing, Solomonoff comes to mind as well. For example, Solomonoff's Universal Theory of Prediction, which forms the theoretical foundations for machine learning and is closely linked to Bayesian theory. An interesting overview of this line of work can be found in [120–122].

3.4 The Bayesian view

It is important to remark that an alternative way exists to tackle the surrogate modeling problem. Namely one rooted in Bayesian statistics. The Bayesian view starts from the premise that the process of constructing $\tilde{f}$ is one full of uncertainty [123] and that this uncertainty should be carefully quantified and taken into account when constructing $\tilde{f}$ or making predictions with $\tilde{f}$. There is a wealth of information available on these topics with most work on Bayesian models and Gaussian Process models. Key authors in this respect are Kenneth, and O'Hagan. Only the basics are sketched here, more information can be found in [123–127] and the seminal work by Jaynes [128].

Instead of the term surrogate or metamodel, the term *emulator* is used. An emulator is a stochastic representation of a deterministic function and distinguished from other types of approximation models (e.g., SVMs) by providing a probabilistic assessment of uncertainty about $f(\cdot)$. The model that is typically used to define $\tilde{f}$ is the Gaussian process model. A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [127].

$f(\cdot)$ is regarded as an unknown function and the uncertainty surrounding it is represented by means of a Gaussian process (for simplicity we only consider the real-valued output case):

$$f(\cdot)|B,\Sigma,\mathbf{r} \backsim N(m(\cdot),c(\cdot,\cdot)\Sigma) \tag{3.9}$$

conditional on hyperparameters B,Σ and $\mathbf{r}$. The notation here means that $\forall \mathbf{x}_1,\mathbf{x}_2 \in \Omega$, $E[f(\mathbf{x}_1)|B,\Sigma,\mathbf{r}] = m(\mathbf{x}_1)$ and $Cov[f(\mathbf{x}_1),f(\mathbf{x}_2)|B,\Sigma,\mathbf{r}] = c(\mathbf{x}_1,\mathbf{x}_2)\Sigma$, where $c(\cdot,\cdot)$ is a positive-definite function such that $\forall \mathbf{x}\; c(\mathbf{x},\mathbf{x}) = 1$. Thus a stationary, separable covariance structure is assumed with the covariance between the outputs given by the positive-definite matrix $\Sigma \in \mathbb{R}^{+}_{q,q}$ and with $c(\cdot,\cdot)$ providing spatial correlation across the input space. Typically the mean and correlation functions are modeled as

$$m(\mathbf{x}_1) = B^T\mathbf{h}(\mathbf{x}_1) \tag{3.10}$$

$$c(\mathbf{x}_1,\mathbf{x}_2) = exp\{-(\mathbf{x}_1-\mathbf{x}_2)^T R(\mathbf{x}_1-\mathbf{x}_2)\} \tag{3.11}$$

with $\mathbf{h} : \Omega \mapsto \mathbb{R}^m$ an arbitrary vector of m regression functions $h_1(\mathbf{x}),...,h_m(\mathbf{x})$ shared by each output $f_j(\cdot)$, $j = 1,...,q$; $B = [\beta_1,...,\beta_q] \in \mathbb{R}_{m,q}$ a matrix of regression coefficients; and R a diagonal matrix of p positive roughness parameters $\mathbf{r} = (r_1,...,r_d)$ (the correlation parameters). For the prior mean structure $m(\mathbf{x})$ a linear regression has been found to be adequate in most applications but higher-order polynomials may be used as well. A Gaussian correlation function for $c(\cdot,\cdot)$ is also typical [129]. To complete the model specification priors must be selected for the unknown hyperparameters $\Sigma,B,\mathbf{r}$. Conventionally 'non-informative' priors are used if little other evidence is available. Once the model is specified, standard Bayesian inference methods can be used to perform prediction and update the model with new data. For details please refer to the references above.

This Bayesian approach has a number of advantages. The most important being that every bit of uncertainty is quantified through the priors and taken into account throughout the inference. This gives more confidence in the model since it is always clear to what degree the model can be trusted. This also allows for a natural formulation of an adaptive sampling procedure: select data points in those regions where the uncertainty is reduced the most. The use of priors also allows one to incorporate prior knowledge there may be about the expected behavior of the response. Finally, another advantage is that the final result of the model construction procedure is not a single, unique solution but rather a distribution over all possible models.

The majority of research in this area has been conducted using Gaussian Processes. Gaussian process models are very closely related to Gaussian Neural Networks, Kriging models, RBF models, and SVM models. An excellent treatment of the theory and links between these different model types can be foud in [127]. We will not discuss the Bayesian approach any further in depth since it warrants a dissertation in its own right.

3.5 Surrogate modeling requirements

3.5.1 Reference model

In order to apply surrogate modeling methods, obviously some kind of reference model (also referred to as a simulator, object model, or disciplinary model) is needed. Be it in the form of a simulation code and its dependencies, a set of equations, or a pre-generated dataset. The more information available about the reference model the better, this is where interaction with the domain experts is paramount. Useful information includes:

- state of the implementation (stable, under development, ...) and usage requirements (including license information)
- system type (deterministic, stochastic, dynamic, ...)
- availability of existing approximation models and their restrictions
- dimensionality, domain, and type (real, complex or discrete) of the inputs and outputs
- potential for missing or *don't care* values and estimation of the noise level/type
- any information about discontinuities, non-linearities, sensitive parameters, epistasis, non-excited modes (sleeping dynamics), feedbacks, etc.

Remember that, *in principle* one should be able to use the reference model directly. However for practical reasons an explicit choice is made to use an approximation model instead. Therefore, if the quality of the reference model is inadequate for the task at hand one should seriously question the usefulness of an approximation model based on the reference model (Garbage-In-Garbage-Out).

3.5.2 Model requirements

Besides the reference model, arguably the most important aspect of surrogate modeling is knowing what minimum requirements the surrogate has to meet in order for it to be useful within the final application. Defining these requirements turns out to be much more challenging than one would expect at first. We shall revisit this topic in chapter 8. For now it suffices to give a list of possible questions that should be asked:

- What software, time, budget, expertise, ... is available for building and testing the surrogate model. For example, [130] report 152.6 hours necessary to construct a RBF neural network versus 3 minutes for a regression tree on the same dataset. Depending on the situation this may or may not be a problem.
- How will the quality of the metamodel be assessed (formal analysis, benchmark scenarios, etc.).
- What level of traceability is required? What process knowledge, physics, parameter interactions, etc. should be recognizable in the final metamodel. This also includes things like adhering to documentation and rapportation protocols.
- What will the surrogate mainly be used for: interpolation, extrapolation, validation, etc.. Note that using a purely data-based surrogate model outside of the domain it was designed for is pure speculation.
- What deviations with respect to the reference model are acceptable and how will they be measured (accuracy).
- Should the surrogate model be able to associate any prediction with the uncertainty of that prediction
- What are the resource restrictions on the final surrogate (execution speed, memory usage, ...). For example, this could be an issue if the model is integrated into a hardware controller or software package.
- Should the metamodel be able to interoperate easily with other systems (expert system, database, ...)
- Will the metamodel be used within a chain of other metamodels. If so, what are the requirements on data scaling/format, software platform, etc.

As always, the answers to these questions will depend completely on the problem and target application. The process of building an approximation model for use in missile control will be completely different than one built for data visualization.

3.6 Data collection strategy

If the decision is made to construct a surrogate model for a given data source, one of the first questions that arises is what algorithm to use to query the data source. The relevant theory is referred to as experimental design, or, Design of Experiments (DoE) and dates back to work by Fisher [131] in the context of designing *physical* experiments.

3.6.1 Classic "one-shot" experimental design

The goal of DoE is to determine the optimal set of experiments to perform (from a given parameter space) such that they are maximally informative (according to one or more criteria) given a limited budget. The theory was initially developed for physical experiments, thus the proper treatment of measurement error played a major role. According to Fisher important principles of experimental design are: randomization (of the sample population), replication (to deal with measurement error), blocking (grouping experimental units to reduce variability), orthogonality, and comparison. Examples of classic designs are fractional factorial, Plackett–Burman, split plot, and nested designs.

These designs were later adapted, improved, and new designs formulated with the development of response surface methodology in the early 1950's by Box and Wilson [82] (e.g., Box-Bhenken and Central Composite designs). The methodology became very successful, in 1971 Myers notes the "*successful application of known RSM techniques in such areas as chemistry, engineering, biology, agronomy, textiles, the food industry, education, psychology, and others*" [132, 133]. This work on RSM designs eventually led to the development of so-called optimal designs, a very important and widely used class of methods.

The concept of optimal design was pioneered by the Finnish mathematician Gustav Elfving in the early 1950's [134] and the work by Kiefer and Wolfowitz in the early 1960's [135]. The adjective optimal refers to the fact that the distribution of points is such that they are optimal with respect to some statistical criterion and metamodel (e.g., a second order polynomial). In the design of experiments for estimating statistical models, optimal designs allow model parameters to be estimated without bias and with minimum-variance. A non-optimal design requires a greater number of experimental runs to estimate the parameters with the same precision as an optimal design. The optimality of a design depends on the statistical model and is assessed with respect to a statistical criterion, which is related to the variance-matrix of the estimator. Classic optimality criteria include A-optimality, D-optimality, and E-optimality. The letters A,D,E referring to which matrix operator is used to measure optimality given the information matrix X^TX of the design (A=average or trace, D=determinant, E=eigenvalue). If the type of statistical model is not assumed to be known (e.g., Kriging only assumes smoothness) then space-filling may become the criterion (see below).

A vast amount of literature is available from the statistics community on these topics and the resulting methods are used in fields ranging from agriculture to chemistry and medicine (e.g., optimal subject selection in clinical trials). With the rise of computer-based experimentation (cfr. section 2.3) these methods were adapted and further extended to the virtual domain that is computer simulation. The main works in this respect being that by Sacks et. al. [6] and Kleijnen [8]. This evolution brought about a major change in mindset: with computer experiments data is deterministic and noise free. There certainly are stochastic codes, but the main focus of computer experimentation has been on deterministic codes. Thus, in computer experiments there is no need to do replicates (as opposed to physical experiments) and the classic theory of optimal design is not useful. Instead, a major concern is to create an experimental design which can sample the complete design space in a representative way with a minimum number of samples [136]. In this sense popular methods are orthogonal arrays and Latin Hypercube Sampling (LHS)[2]. A good overview of this line of work can be found in [138, 139] and the references therein.

3.6.2 Sequential experimental design

There are a number of drawbacks to the use of static experimental designs in the context of this thesis. First, static designs require the number of points to be chosen upfront. What if this number proves to be too low? How/where should more points be added? Given the iterative nature of the design process it is therefore natural to consider a sequential, or iterative DoE. An algorithm is defined that sequentially selects and collects data points until some threshold is reached, preventing the expensive cost of over-sampling (cfr. section 3.2). Many sequential algorithms have been defined, including *iterative formulations of classic DoE methods*.

A first method worth mentioning are designs that are generated from minimum discrepancy sequences. Minimum descrepancy sequences were originally conceived to develop space-filling points, primarily for the purpose of efficient numerical integration of multidimensional functions. Popular examples are Hammersley, Halton, Sobol and Faure sequences. These methods are also referred to as *quasi-Monte Carlo methods* since they generate a deterministic sequence of points. This makes them useful for incremental model building [4].

There is of course also the Bayesian approach [140]. In the Bayesian framework, a prior distribution over the space of all possible functions from inputs to outputs is set. Given available data, a posterior distribution is generated from which new points can be drawn. This method extends naturally to incorporate measurement and prediction of derivatives, partial derivatives and definite integrals of the function [9]. The difficulty, as is typically the case with Bayesian methods, is a suitable formulation of the prior. Examples of Bayesian experimental designs include Mean Squared Error designs, Minimax Designs, and Maximum Entropy Sampling [23].

In general the sequential design problem can be seen as an iterative optimization process driven by one or more criteria (section 3.2). Possible criteria include:

- distance from neighboring points (space-filling)
- non-linearity of the response
- uncertainty of the surrogate model in use

[2]Latin Hypercube sampling grew out of research into risk analysis, see [137].

- distance from the (estimated) optima
- degree of constraint violation (if any)

Depending on the surrogate modeling goal (optimization versus global model) the subset of criteria used will vary (note that multiple criteria may be used in concert). The fundamental choice being whether to focus on

- **exploration**: selecting points in regions of which little or no information, or
- **exploitation**: selecting points in the most promising regions.

Remark the similarity with the Bias-Variance trade-off problem from machine learning (section 3.7.2) and remark that this is, at heart, again an optimization problem with different solution strategies (see for example [95]).

The second criteria above ("select proportionally more points in highly-nonlinear regions of the response") is the one most often used by authors when proposing new sampling algorithms. However, applying it implicates an assumption that is not always true, often overlooked, and worth emphasizing here. The assumption is that the nonlinear regions are the most difficult to fit and thus require proportionally more data. While intuitively appealing the assumption is not always valid. For example Kriging, and other kernel-based models, often have little problem capturing complex non-linearities but have much more difficulty with the smoother, flatter regions of the response. Furthermore, data clustering induced by focusing too much on nonlinear regions can cause numerical problems [141]. The goal of sequential design strategies is to improve the accuracy of the response model and reduce the uncertainty associated with its use. While the region that is deemed most interesting by these two criteria may coincide with the most nonlinear region, this is definitely not always the case. More discussion and concrete examples may be found in [9, 142, 143].

This remark aside, some authors also make the explicit distinction between passive and active adaptive sampling. E.g., Gautier et al. [144–146] do so from a systems control perspective:

- **passive sequential design**: each step in the sequential design is considered as the last one to be performed
- **active sequential design**: takes into account the fact that further observations will be available when tuning the model

A rich variety of adaptive sampling methods have been proposed [12, 14, 23, 115, 138, 147–156]. A promising area of current research is the combination of multiple sample selection criteria in a dynamic manner, e.g., as done in [157]. This point will be revisited in section 4.5.2.

3.6.3 Remarks

Three important remarks can be made with regard to the data collection strategy. First, in the two preceding sections we have placed the focus on data collection in the input space. However, data collection requirements may needed on the output space as well. For example an application may require that a certain output band is more densely sampled relative to the remaining range.

Second, as should be clear from the discussion, we are concerned with the case where the data source is an unknown function that should be queried. For completeness, sometimes this is not the case but instead pre-calculated data is available in the form of a fixed dataset. In these cases the question arises if and how the data should be sub-sampled (*compressed*) in order to make it more *balanced* (ensure a particular coverage distribution of the input space). These problems are more related to data mining and local learning methods (section 3.7.5) than the surrogate modeling algorithms that are the topic of this thesis. However, a good overview of motivation and algorithms involved is given in chapter two of [158].

We conclude with a critical remark with respect to experimental designs that is best summed up by [159]:

> *The results show that there are statistically significant differences between the approximation results of employing different designs, but more often the difference is not significant. In most cases, the number of runs or the sample size has stronger impact on the accuracy than do different designs. When the dimension is low, a small size increment can often reduce more error than do "better designs." To get the desired precision by one-stage method, enough samples may be needed regardless what design is used. Sample size determination may need much more attention for computer experiments.*

Though the quote only refers to static, one-shot designs and is unclear about higher dimensional cases, it is an important reminder that an adaptive, 'intelligent' approach does not automatically guarantee significant time savings and improved results. The author is unfortunately unaware of analogous work in the case of sequential design.

3.7 Modeling strategy

Some kind of executable mathematical model is needed to interpolate and extrapolate between the raw data. This again involves a number of choices, detailed in the following subsections.

3.7.1 Model type

The first obvious problem is selecting the surrogate model type and the model parameter optimization method (cfr. the two minimizations in equation 3.2). Popular surrogate model types include Radial Basis Function models, Rational Functions, Artificial Neural Networks, Support Vector Machines, Multivariate Adaptive Regression Splines, and Kriging models.

Proper choice of the model type is important and depends on the required (see also [5])

- interpretability
- adaptability
- target implementation platform
- resource constraints
- amount of available data
- prediction speed
- composability (will there be multiple models that need to be combined)
- traceability (how well can relationships, behavior, dynamics, ... explicitly be traced back to the reference model)
- portability (how easy is it to port the surrogate model to different languages and platforms)

Different model types are preferred in different domains. For example rational functions are widely used by the Electro-Magnetic (EM) community [160, 161], while ANNs are preferred for hydrological modeling [67]. Differences in model type usage are mainly due to practical reasons (available expertise, tradition, computation time, etc.), though there are some exceptions: In some cases the choice of the metamodel type can also be motivated by knowledge of the underlying physics[3] [162] or by the special features the model provides: for example the uncertainty prediction based on random process assumption in Kriging methods[4] [163].

However, in general there is no hard theory that can be used as an a priori guide, thus claims that a particular model type is superior to others should always be taken

[3]Knowledge of the physics of the underlying system can make a particular model type to be preferred. For example, rational functions are popular for all kinds of Linear Time-Invariant (LTI) systems since theory is available that can be used to prove that rational pole-residue models conserve certain physical quantities (see [162]).

[4]Kriging models are closely related to Gaussian Process (GP) models and often Kriging and GP models are used as labels for the same techniques. Great similarities between GP models, SVM models, RBF models, and RBF Neural Networks exist as well, as has been discussed in [127].

with a grain of salt. This is related to the so called *No-Free-Lunch-Theorems* [164]. One of the many formulations is as follows[5]:

> *...it is impossible to justify a correlation between reproduction of a training set and generalization error of the training set using only a priori reasoning. As a result, the use in the real world of any generalizer which fits a hypothesis function to a training set (e.g., the use of back-propagation) is implicitly predicated on an assumption about the physical universe.* [165]

Basically the theorems explain why, over the set of all possible learning problems, each algorithm will do on average as well as any other due to the bias in each algorithm.

We shall revisit this point in much more detail in chapter 7.

3.7.2 Hyperparameter identification

Selecting the model type (optimization over T in equation 3.2) is only part of the problem. Once the model type has been fixed, the complexity of the model needs to be chosen as well (optimization over θ). Each model type has a set of model parameter θ (hyperparameters) that control the complexity of the model and thus the bias-variance trade-off.

The bias-variance trade-off [166] is one of the most important concepts in machine learning and worth re-stating here. The trade-off helps explain why there is no universally optimal learning method. Its derivation is usually given for the Mean Square Error (MSE) but derivations for other loss functions exist as well.

Let D be a set of noisy training data generated from a true function $y = f(\mathbf{x})$ and let $\tilde{y} = \tilde{f}(\mathbf{x})$ be a model that fits the data. So $D = \{(\mathbf{x}_1, t_1), ..., (\mathbf{x}_k, t_k)\}$ with $t_i = y + v$ and $E\{v\} = 0$. A natural way to assess the quality of the model is through the MSE with respect to an infinite test set (hence the use of the expectation):

$$E\{MSE\} = E\left\{\frac{1}{k}\sum_{i=1}^{k}(t_i - \tilde{y}_i)^2\right\} = \frac{1}{k}\sum_{i=1}^{k}E\left\{(t_i - \tilde{y}_i)^2\right\} \tag{3.12}$$

Working out the last sum gives:

[5]See also http://www.no-free-lunch.org/.

$$\begin{aligned}
E\left\{(t_i-\tilde{y}_i)^2\right\} &= E\left\{(t_i-y_i+y_i-\tilde{y}_i)^2\right\}\\
&= E\left\{(t_i-y_i)^2\right\}+E\left\{(y_i-\tilde{y}_i)^2\right\}+2E\left\{(y_i-\tilde{y}_i)(t_i-y_i)\right\}\\
&= E\left\{\nu^2\right\}+E\left\{(y_i-\tilde{y}_i)^2\right\}+2(E\{y_it_i\}-E\left\{y_i^2\right\}-E\{\tilde{y}_it_i\}+E\left\{\tilde{y}_iy_i\right)\}\\
&= E\left\{\nu^2\right\}+E\left\{(y_i-\tilde{y}_i)^2\right\}\\
\textit{Note} &: E\left\{y_it_i\right\}=y_i^2 \text{ since } f \text{ is deterministic and } E\left\{t_i\right\}=y_i\\
&: E\left\{y_i^2\right\}=y_i^2 \text{ since } f \text{ is deterministic}\\
&: E\left\{\tilde{y}_it_i\right\}=E\left\{\tilde{y}_i(y_i+\nu)\right\}=E\left\{\tilde{y}_iy_i+\tilde{y}_i\nu\right\}=E\left\{\tilde{y}_iy_i\right\}+0
\end{aligned}$$

since the noise in the infinite test set over which the expectation is probabilistically independent of the model $\tilde{f}$

Thus, the MSE can be decomposed in expectation into the variance of the noise and the MSE between the true and predicted values. Further decomposition gives (recall that the bias of an estimator $\hat{\theta}$ for θ is $bias\{\hat{\theta}\}=E\{\hat{\theta}-\theta\}$):

$$\begin{aligned}
E\left\{(y_i-\tilde{y}_i)^2\right\} &= E\left\{(y_i-E\{\tilde{y}_i\}+E\{\tilde{y}_i\}-\tilde{y}_i)^2\right\}\\
&= E\left\{(y_i-E\{\tilde{y}_i\})^2\right\}+E\left\{(E\{\tilde{y}_i\}-\tilde{y}_i)^2\right\}+2E\left\{(E\{\tilde{y}_i\}-\tilde{y}_i)(t_i-E\{\tilde{y}_i\})\right.\\
&= bias^2+Var\{\tilde{y}_i\}+2(E\{y_iE\{\tilde{y}_i\}\}-E\left\{E\{\tilde{y}_i\}^2\right\}-E\{\tilde{y}_iy_i\}+E\left\{\tilde{y}_iE\{\tilde{y}_i\right.\\
&= bias^2+Var\{\tilde{y}_i\}\\
\textit{Note} &: E\{y_iE\{\tilde{y}_i\}\} \text{ since } f \text{ is deterministic and } E\left\{E\{z\}\right\}=z\\
&: E\left\{E\{\tilde{y}_i\}^2\right\}=E\{\tilde{y}_i\}^2\\
&: E\left\{\tilde{y}_it_i\right\}=E\left\{\tilde{y}_i(y_i+\nu)\right\}=E\left\{\tilde{y}_iy_i+\tilde{y}_i\nu\right\}=E\left\{\tilde{y}_iy_i\right\}+0\\
&: E\{\tilde{y}_iy_i\}=y_iE\{\tilde{y}_i\}\\
&: E\left\{\tilde{y}_iE\{\tilde{y}_i\})\right\}=E\left\{\tilde{y}_i\right\}^2
\end{aligned}$$

Thus the final decomposition of MSE in expectation becomes:

$$E\left\{(t_i-\tilde{y}_i)^2\right\}=Var\{noise\}+bias^2+Var\{\tilde{y}_i\} \tag{3.13}$$

The optimal model $\tilde{f}$ (with respect to the MSE loss function) is the one that minimizes this expression. Since the variance of the noise is fixed, in order to minimize the MSE both the variance and bias must be minimized. However, this is not trivial since this is a trade-off. Take the two extremes for example. $Var\{\tilde{y}_i\}$ can be minimized by using a model that always predicts a constant value. In that case the model prediction is independent of the data ($Var\{\tilde{y}_i\}=0$). However, the amount we are off the real function (the bias) would be huge. On the other hand, if the model perfectly interpolates the data ($bias^2=0$ since $E\{\tilde{y}\}=y$) the variance term will become equal to the variance of the noise, which may be significant. The model becomes very sensitive to the data distribution (high variance), a process typically referred to as overfitting.

This relates to the complexity as follows: Models with too few parameters are inaccurate because of a large bias (not enough flexibility) while models with too many parameters are inaccurate because of a large variance (too much sensitivity to the data). Thus identifying the best model requires identifying the proper model complexity (number of parameters).

The complexity of the model is thus controlled by a set of hyperparameters. For example, with rational functions this could be the degree of the nominator and denominator monomials, with Kriging models the correlation/regression functions and associated parameters, with ANN the number of units per hidden layer, and with SVM the kernel function, the regularization constant γ, and the kernel parameters (e.g., the spread σ in the case of an RBF kernel).

Classically, this is done based on expert domain knowledge (e.g., setting the Kriging trend function) or through a trial-and-error procedure (see for example [167, 168]). However, in essence this is an optimization problem in the space of possible models. Thus, a better approach is to use an optimization algorithm guided by a performance metric (e.g., external validation set, leave-one-out error, an approximation of the posterior $P(\theta, M|data)$ such as AIC, etc.). In this way a successive set of approximation models are generated that converge towards a local minimum of the optimization landscape, as determined by the performance metric. An advanced example in this respect is given in [169]. The challenge here is to converge to a good optimum, since the landscape can be expected to be highly multi-modal, deceptive and contain dependencies between variables (epistasis). This makes the choice of the performance metric crucial, leading us to the well known problem of model selection and accurate generalization estimation. This is discussed in the next section.

Note, though, that in the context of this work the hyperparameter optimization problem is more difficult than is typically the case in machine learning. Since data points are costly, the use of sequential design is unavoidable (cfr. section 3.6). This implies that the data distribution (from which models must be constructed) is not constant, and consequently that the hyperparameter optimization surface is dynamic instead of static (as is often assumed). These points will be revisited in section 4.5.4.6 and chapter 7.

3.7.3 Model selection

The most crucial aspect of searching through the space of possible models is selecting an accurate criteria to guide this search. In his book *The Symmetric Eigenvalue Problem* the mathematician B. N. Parlett remarks "*There is little profit in approximations which are good but not known to be good*". This is the classical model selection problem.

Many model selection criteria have been developed: empirical methods based on resampling (e.g., cross validation, leave-one-out, bootstrapping) [170], information theoretic methods based on Bayesian statistics (e.g., AIC) [127, 171–177], or metrics with their roots in Statistical Learning Theory (e.g., VC-dimension [178–180]).

Again, each metric has its strengths and drawbacks. By far the most popular are data-based empirical methods [181–184] since these are easiest to implement and apply generically. However, being data-based they can be misleading if data is sparse of the distribution sub-optimal (e.g., clustered) [9]. Another disadvantage is that they are generally computationally expensive to use since they require retraining the model multiple times. However, this cost can be alleviated in some model types through the use of mathematical shortcuts. See for example the discussion on linear regression in [8].

In general the choice of model selection metric depends completely on the characteristics the domain expert would like to see in the model. This can include:

- accuracy
- smoothness
- bounded output range
- low complexity
- etc.

However, mapping domain expert requirements onto concrete model selection criteria is not as obvious as may seem at first. This is discussed in depth in chapter 8. A wide range of references are available that discuss and compare different model selection criteria [11, 170, 171, 181, 185–191]. Another good resource is the collection of links at `http://www.modelselection.org/`.

3.7.4 Hierarchical modeling and Ensembles

Often the systems to model are highly non-linear and may contain discontinuities. Trying to capture the full behavior of the system in a single, global model may prove to be too difficult. There are different ways this can be dealt with through the use of hierarchical modeling[6].

3.7.4.1 System level decomposition

The first solution is a domain specific one. If the problem to be modeled is sufficiently modular it can be decomposed into smaller components and each modeled separately.

[6]This should not be confused with multilevel modeling which is a generalization of linear and generalized linear modeling in which regression coefficients themselves are given a model, whose parameters are also estimated from the data [192, 193].

A different modeling setup can be used for each component and once generated all models can be chained together to approximate the full system. Though care must be taken to decompose the system in such a way that the error and uncertainty propagation through the model network is kept under control.

3.7.4.2 Input space level decomposition

An alternative approach of decomposing the modeling problem is by using domain knowledge (or screening techniques if little knowledge is available) to group sets of related input variables together. Variables may be grouped in a hierarchical or non-hierarchical manner. This line of work can be traced back to the rigorous work by Kron and his work on the development of diakoptics [194]. Diakoptics is a method for finding the solution to large scale systems in a piecewise fashion [195]. The word diakoptics is derived the Greek word *kopto* which means *to tear* and *dia* which can be interpreted as system [195]. Hence diakoptics is the method of system tearing.

This kind of decomposition is based on the assumptions of effect sparsity (not all factors are relevant) [196]. Each subgroup of variables is analyzed independently and the data combined to form the final surrogate model. These methods assume that from the total set of design variables for the system, one can subdivide the set into disjoint subgroups and that the interactions between individual design variables in disjoint subgroups are of negligible importance to the overall variability to the response. More information about these techniques can be found in [197].

Alternatively, instead of grouping different variables together, one can retain all variables but apply a divide-and-conquer strategy on the input space. The input space is partitioned recursively until a model with acceptable performance can be built on each partition. The overall surrogate model then consists of a tessellation of simpler sub-models. The advantage of this approach is that the model for each partition is only as complex as the data needs it to be. For example, in the case of polynomial models, plateaus will be modeled by simple one degree polynomials while more nonlinear partitions will be fitted with higher degree polynomials. While intuitively very attractive there are a number difficulties. The first is designing a good partitioning algorithm. For rectangular partitions this can be done based on a clustering analysis of the data or based on derived gradient information. Alternatively, model uncertainty information can be used to delineate partitions. However, for non-rectangular partitions things rapidly become more complex. Another problem is that the number (and size) of partitions grows exponentially with the dimensionality of the input space. This makes that constructing separate models for each partition (including finding a good model complexity) quickly becomes computationally prohibitive for reasonable problems. Finally there is the problem of ensuring continuity at partition boundaries, meaning some form of overlap must be defined. These problems aside, there have been a number of successful applications [198–201].

3.7.4.3 Ensembles

Ensemble methods are closely related to divide-and-conquer modeling and have been very popular within the neural network literature (cfr. committee networks) [202]. An ensemble is simply a group of more than one model whose predictions are combined such that the group acts as a single model. Under certain conditions it can be proven that by combining models in this way a superior prediction may be achieved [203,204]. There is a huge amount of literature available on ensemble methods [205–217].

The many different types of ensembles are distinguished on the basis of how group members are constructed and combined. Each ensemble member may be of the same type and complexity but trained on different portions of the data. This can be useful when combined with cross-validation in order to estimate the prediction uncertainty. On the other hand, the ensemble members may be heterogeneous but share the same data, again useful for prediction variance estimation. In general, the difficulty is choosing ensemble members in such a way that they are optimally complementary (the more the ensemble members are aligned, the less information combining them gives). A popular algorithm that helps achieve this is Negative Correlation Learning [207].

Besides the composition of the ensemble, a problem is also how to combine the different models. This may be done in a hierarchical manner, through weighted averaging, voting, or any other aggregation function. While ensembles have seen many successful uses, and are increasingly being used in optimization [18, 87], the main problem with their application is that they introduce many new parameters, design choices, and increase the computational cost of the modeling.

3.7.5 Local or lazy learning

A completely different way of dealing with the data approximation problem is through the use of local learning or lazy learning, excellent reviews of which are given in [218–220]. Traditional approximation methods are constructed based on all available data in a training phase. Once the model parameters are determined the model can then be used for prediction and the training data is no longer needed. These global models stand in contrast to local approximation methods like nearest neighbor, Locally Weighted Regression (LWR) or Moving Least Squares [221]. Local regression is an old method for smoothing data and has its origins in the graduation of mortality data and the smoothing of time series in the late 19th / early 20th century.

Local models only construct a model once a prediction for an unknown point is requested, hence the term lazy. Given an unknown point, a local learner will determine the relevant neighbourhood for that point and construct a model for that neighborhood only. Usually the neighborhood points are weighted with respect to the distance to the query points[7].

[7]Readers may note the relationship with kernel methods like Kriging, SVM, and RBF.

Local methods differ in: (1) the neighbourhood size (*bandwidth*), (2) the weight assignment function, (3) the parametric model family that is fitted locally, and (4) the assumptions about the distribution of the response. Each of these may vary with the domain or change adaptively. A classic method is LOESS, or LOWESS (locally weighted scatterplot smoothing) [219]: At each point in the data set a low-degree polynomial is fitted to nearby data points. The polynomial is fitted using weighted least squares, giving more weight to points near the query point.

A new, more powerful local learning method that was developed recently is Locally Weighted Projection Regression (LWPR) [220, 222]. At its core, LWPR uses locally linear models, spanned by a small number of univariate regressions in selected directions in the input space. According to the authors it learns rapidly with second order learning methods based on incremental training (adaptive sampling), has a computational complexity that is linear in the number of inputs, and can deal with a large number of (possibly redundant & irrelevant) inputs.

Local methods are very useful in the case of large data sets and they scale well with the dimensionality. They also relieve the modeller from having to decide upfront about a global model structure. Additionally, being local makes them robust with respect to heterogeneous response behaviors. Predictions in a relatively flat area of the response are not impacted by strong nonlinearities in a different part of the response (in contrast to global models like polynomials). They are, however, slower to use since the local model must be constructed for each prediction. This can be alleviated somewhat through an efficient implementation (e.g., pre-calculate neighbourhoods where possible) and if the model structure is kept relatively simple. Another disadvantage with respect to global models is that it is hard to write a closed form expression of the model. Transferring the model usually means transferring the data and the procedure to generate the model.

3.7.6 Knowledge based modeling

A last way in which the modeling process may be enhanced is through the incorporation of knowledge. Often some information is available about the expected response behavior (e.g., monotonicity [223]) and such information should be included in the model. As discussed in section 2.5, pure data based methods cannot be trusted to extrapolate or interpolate in regions where data is missing or very sparse. Unless the chosen model structure is known to be representative of the true system of course. There are different ways existing process knowledge (in the form of one or more analytical formulas) can be incorporated into surrogate models (we ignore MOR approaches as discussed in section 2.5.1). The first is by using a data based approximation method that is directly inspired by the application domain. An example is the Vector Fitting (VF) technique in use in EM applications [224]. A second way is by incorporating the knowledge into the structure of a generic fitting method. With Kriging models this may be done through the use of a domain-inspired regression function. In neural networks this may be done by embedding custom transfer functions or using Knowledge Based Neural Networks [24, 225].

A different, popular, way of including domain specific knowledge in a surrogate model is through the use of (space) mapping methods [76]. The core idea here is to take the available knowledge as a starting point and use generic approximators to map between the true system and the approximate knowledge. There are three main types of mapping methods which we discuss briefly below.

Let $\mathbf{x}$ and $\mathbf{y}$ be vectors of model inputs and outputs. Let the existing knowledge be represented by function $\mathbf{y} = f_{emp}(\mathbf{x})$, which can be either empirical functions or circuit formula. Let the black-box approximation method be represented by $\tilde{\mathbf{y}} = \tilde{f}(\mathbf{x})$. The problem is then to determine how the two parts $f_{emp}(\mathbf{x})$ and $\tilde{f}(\mathbf{x})$ should interact such that their combination results in superior accuracy than if each were used in isolation. There are different ways to accomplish this.

The first is through Difference Mapping (DM) [74]. In this case $\tilde{f}(\mathbf{x})$ will be trained to learn the difference between the true data $\mathbf{y}$ and the approximation $\tilde{\mathbf{y}}$ given by $f_{emp}(\mathbf{x})$. Thus the true output $\mathbf{y}$ is written as

$$\mathbf{y} = \tilde{\mathbf{y}} + \Delta\mathbf{y} = f_{emp}(\mathbf{x}) + \tilde{f}(\mathbf{x}) \tag{3.14}$$

This method is expected to give good results when the difference has a simpler input-output relationship as a function of the inputs than the target data. Note that the idea here is similar to the fundamental structure of Kriging and Gaussian Process models. One could interpret those models as a base model that is fitted to the data (the regression) and is then corrected by the correlation function.

A second method is Output Mapping (OM), Prior Knowledge Input (PKI) [74], or Manifold Mapping [226, 227]. In this case the outputs of the existing empirical model $\tilde{\mathbf{y}}$ are used as inputs to the black-box model, in addition to the original problem inputs $\mathbf{x}$. Thus the black-box model is trained on $(\mathbf{x}', \mathbf{y})$ with $\mathbf{x}' = (\mathbf{x}, \tilde{\mathbf{y}})$, a simpler input-output relationship as compared to the original problem, which requires less training data [228]. The target data is now predicted as

$$\mathbf{y} = \tilde{f}(\mathbf{x}') = \tilde{f}(\mathbf{x}, f_{emp}(\mathbf{x})) \tag{3.15}$$

The third mapping function is Input mapping (IM) or Space Mapping (SM) [229] and is probably the best known. IM intelligently links companion "coarse" (ideal, fast, or low fidelity) and "fine" (accurate, practical, or high-fidelity) models of different complexities, e.g., empirical circuit-theory based simulations and full-wave EM simulations, to accelerate iterative design optimization. Through IM optimization, the surrogates of the fine models are iteratively refined to achieve the accuracy of EM simulations with the speed of circuit-theory based simulations.

We adopt the notation of [229]. Let vectors $\mathbf{x}_c$ and $\mathbf{x}_f$ represent the design parameters of the coarse and fine models, respectively. Let $R_c(\mathbf{x}_c)$ and $R_f(\mathbf{x}_f)$ represent the corresponding responses of the coarse and fine models, respectively. The response of the coarse model R_c is much faster to calculate, but less accurate than the response of the fine model R_f. The aim of space mapping optimization is to find an approximate mapping P from the fine model parameter space $\mathbf{x}_f$ to the coarse model parameter space $\mathbf{x}_c$, i.e., $\mathbf{x}_c = P(\mathbf{x}_f)$ such that $R_c(P(\mathbf{x}_f)) \approx R_f(\mathbf{x}_f)$ [76]. P is usually implemented using one of the available black-box fitting methods (e.g., ANNs as done in [230]). Thus the final output is calculated as

$$y = R_c(\mathbf{x}_c) = R_c(P(\mathbf{x}_f)) = R_c(\tilde{f}(\mathbf{x})) \tag{3.16}$$

In particular space-mapped neuromodeling has demonstrated its efficiency by passive modeling or small-signal device modeling, achieving fast and accurate models for devices such as bends, high temperature superconductor filters, embedded passives in multilayer printed circuits, and other linear components [231].

In sum, a wide range of knowledge incorporation methods have been developed and these should always be preferred over pure black-box methods. The disadvantage is that these approaches only work on problems where the underlying process is at least partially known and knowledge can be encapsulated into standalone rules and/or equations.

3.7.7 Hybrid modeling

Note that the methods described in the preceding subsections should not be seen as independent algorithms but more as a continuous scale where one morphs into the other. There are many variations on the main points given above and all can be combined in endless ways. For example [43] and others [44] discusses a surrogate modeling strategy where models are built on a mixture of low-fidelity and high-fidelity simulations which does not fit directly in one of the previous categories. This set of methods are typically referred to as multi-level or multi-fidelity approimations.

In the limit one can visualize a web linking different model types, divide-and-conquer schemes, and space mapping methods. This web may take the form of a complex, hierarchical ensemble, or even a neural network where each neuron is in itself a full blown model. While at it, why not evolve the whole web using an evolutionary algorithm to determine the optimal topology?

While interesting from a research perspective in general the downside of more esoteric hybrid methods is that they are very computationally expensive and demand a lot from the designer with respect to implementation and tuning complexity. Furthermore it still remains to be seen whether the performance of an ultra adaptive, hybrid approach performs marginally better than one properly applied classical statistical model. Also, as [4] states, "*...it must also be remembered that even the most complex approaches are still subject to the "no free lunch" theorems.*

3.8 Other surrogate modeling choices

So far we have concentrated on the two major aspects of applying surrogate modeling: data collection and modeling strategy. However, besides these surrogate modeling also involves making informed choices with respect to: feature selection, data pre/post-processing, termination criteria, and model validation. In particular, feature selection is gaining increasing importance as the number of dimensions considered by engineers continues to increase. One of the biggest challenges currently facing surrogate modeling researchers is how to identify unimportant or correlated variables when data is scarce (simulations are costly) and/or when samples cannot be chosen freely (e.g., due to physical constraints).

More of a practical consideration is the sample evaluation strategy to use. When constructing an approximation model for an expensive simulation engine the largest computational bottleneck is performing the necessary simulations. A natural step is to harness the power of a grid, cluster, or cloud to reduce this cost. Integration with a distributed system can occur on multiple levels, this will be discussed further in chapter 6.

3.9 Towards adaptive integration

From the previous sections it should be clear that there is a wide variety of interdependent problems that one encounters when generating an approximation model. The optimal solution for each case depending on the problem characteristics: input/output dimensionality, amount of data that can be collected, application constraints, available process knowledge, etc. A huge amount of comparative studies have been conducted that try to extract general rules and guidelines with regard to modeling and sampling strategy choices [2, 3, 10–21]. However, given the complexity of the problem, care must be taken when interpreting such rankings. The goal of this thesis is to explore how one can go beyond comparisons on individual sub-problems and work towards an adaptive integration of sampling, model generation, and simulator scheduling while still allowing for problem specific customizations.

The motivation for this goal is two-fold. First, given the interdependence of the different sub-problems, an adaptive integration of modeling and sampling strategies makes for a more optimal use of available computational resources (a good discussion is given in [9]). The number of simulations can be reduced and the accuracy of the models improved. Similar sentiments have been expressed by other authors [9,22–25]. The second motivation has to do with easing the manual process of generating an accurate surrogate model. Very few domain experts have the time and expertise to deal with the intricacies of different sampling and modeling techniques. Even if they did, readily available software implementations are typically lacking, and papers do not always include enough detail to allow for a custom implementation. Their primary concern is obtaining an accurate replacement metamodel for their problem as fast as possible and with minimal overhead.

At the same time any integration effort should take into account that every discipline or engineer has his preferred technique and approximation method and there is no such thing as a "one size fits all"[8]. Thus any integration should allow for easy switching between methods and should be naturally extensible.

3.10 Conclusion

In chapter 2 we narrowed the scope of the dissertation and explained the basic ideas and evolutions behind surrogate modeling. In this chapter, starting from the domain of surrogate modeling, we explored the different sub-problems and challenges that arise as part of applying surrogate methods: data collection strategy, experimental design, model parameter optimization, etc. The chapter also discussed how there are few objective rules as to what particular combination of ingredients works for which problems. With this chapter all the necessary terms and concepts have been introduced. This foundation can now be built upon to explore the ways the different surrogate modeling ingredients can be incorporated in an adaptive, reusable software framework.

[8]It is true that general purpose approximators exist that work well across many applications. Neural networks being the prime example. Every popular approximation method can be theoretically proven to have universal approximation properties. However, in a practical sense large differences between methods exist with regard to interpretability, sensitivity to data distribution, extrapolation capability, ease of uncertainty estimation, etc.

4 The SUMO Toolbox

Civilization advances by extending the number of important operations which we can perform without thinking about them

– Alfred North Whitehead

4.1 Introduction

With the start of this chapter the foundations and characteristics of surrogate modeling should be clear as well as the challenges this leaves for the domain expert. The goal of the current chapter is to take the discussions from the previous chapter and mould them into a concrete, usable, framework in software. Recall from the introductory chapter that an important motivation for this work was to lower the barrier of entry for domain experts to apply advanced surrogate modeling methods. A readily available software tool would greatly facilitate this. Given the wide range of available methods and problems this is no trivial task. Perhaps a good illustrative quote is one given in the PhD dissertation of Meckesheimer [5] who, in the "future work" section of his concluding chapter, notes:

Developing a framework for metamodel-based design is a multidisciplinary task in itself, encompassing many areas of research in domains ranging from mathematical and statistical modeling and simulation to information technology and systems engineering. The challenge today is to bring these research elements together to exploit their strengths and identify new exciting research opportunities for the future.

This chapter will present such an integrated framework for metamodel-based design and attempt to balance the many trade-offs involved. The framework in question is the SUrrogate MOdeling (SUMO) Matlab Toolbox. The SUMO Toolbox was developed as a common research platform for testing new ideas and methods and it is the platform which is used for all the tests and applications discussed in this dissertation. This chapter shall introduce the toolbox, its history, the philosophy underlying its design, and its software architecture.

Since the topic of this chapter mainly revolves around software it is very prone to quickly becoming dated. While we will focus on the concepts and ideas, it is unavoidable that some aspects will be obsoleted as the software is improved and new versions are released. The most up to date source of information will always be the website which is found at `http://www.sumowiki.intec.ugent.be`.

4.2 Background

4.2.1 History and Motivation

In 2004 research within the Computational Modeling and Simulation (COMS) research group at the University of Antwerp concentrated on developing efficient, adaptive and accurate algorithms for polynomial and rational modeling of electro-magnetic data. On the multivariate front this work resulted in a set of Matlab scripts that were used as a testing ground for new ideas and techniques. Research progressed, and with time (early 2005) these scripts were re-worked and refactored into one coherent Matlab toolbox, tentatively named the Multivariate MetaModeling (M3) Toolbox. The first public release (v2.0) of the M3 Toolbox occurred in November 2006. Development continued but mid 2007 the M3 Toolbox project at the University of Antwerp was discontinued and made available as open source software. The key project members moved to Ghent University in 2007 and there the old code base was resurrected and re-christend as the SUMO Toolbox, developed by the members of the SUMO Lab, at the Integrated Broadband Communications Networks (IBCN) research group. The first public release of the SUMO Toolbox (v5.0) occurred in April 2008. At the time of writing, the current version is 7.0. The motivation for the development of the toolbox has its roots in the electronics domain and a number of factors influenced its instigation.

Firstly there was the observation that, traditionally, surrogate models were created through a manual process where data collection occurred in a one-shot manner according to a straightforward experimental design (usually a factorial). In contrast, research [73] showed that a more intelligent, iterative approach to data collection and model building was more cost effective, particularly since it allowed a form of automation. Some of these early ideas had already found their way into commercial tools (e.g., Agilent Advanced Design System (ADS)) but there were still many unexploited avenues of research with room for increased robustness, automation, and modeling efficiency.

Secondly, given the wide range of choices and options in the surrogate modeling process (cfr. chapter 3) there was a pragmatic need for a common software framework that allowed different ideas and techniques to be easily implemented, tested, and applied to different problems. Given the wide variety of options and methods, such a tool needed to be modular and maximally enable code-re-use and rapid development. Driven by these motivations, the design of the SUMO Toolbox started by taking a step back and formulating the surrogate modeling problem in its most general form. Initially the scope of the project was limited to rational modeling of electromagnetic data. However, decoupling the surrogate modeling problem from the concrete application allowed for a birds-eye view of the modeling process and the implementation of a very flexible and adaptable software stack. As research continued it quickly became clear that the resulting framework was applicable far outside the electronics domain since the essential data fitting problem was the same across many fields.

4.2.2 Design goals

The SUMO Toolbox was designed with a number of goals in mind:

- A flexible tool that integrates different modeling methods and does not tie the user down to one particular set of problems. Reliance on domain specific features should be avoided.

- At the same time keeping in mind that there is no such thing as a 'one-size-fits-all'. Different problems need to be modeled differently and require different a priori process knowledge. Therefore the software should be modular and easily extensible to new (specialized) methods.

- The focus should be on adaptivity, i.e., relieving the burden on the domain expert as much as possible. Given a simulation model, the software should produce an accurate surrogate model with minimal user interaction. This also includes easily integrating with the existing design environment.

- Engineers or domain experts do not tend to trust a black-box system that generates models but is unclear about the reasons why a particular model should be preferred or trusted. Therefore an important design goal was that the expert user should be able to have full manual control over the modeling process if necessary. In addition the toolbox should support fine grain logging and profiling capabilities so its modeling and sampling decisions can be retraced.

Studying these requirements it is obvious that there exist some mutual contradictions. It is impossible to maximize application generality and minimize model accuracy simultaneously. The more knowledge and data you posses about a system, the faster and more accurately you can model it but the less generic your approach will be. We shall revisit this point in section 4.4.

Given this design philosophy, the toolbox can cater to both the researchers working on novel surrogate modeling techniques and to the engineers who need the surrogate model as part of their design process. For the former, the toolbox provides a common platform on which to deploy, test, and compare new modeling algorithms and sampling techniques. For the latter, the software functions as a highly configurable and flexible component to which surrogate model construction can be delegated, easing the burden of the user and enhancing productivity.

4.3 Control flow

A conceptual way of linking the different surrogate modeling components (cfr. section 3.9) together in a control flow is illustrated by the flowchart in figure 4.1.

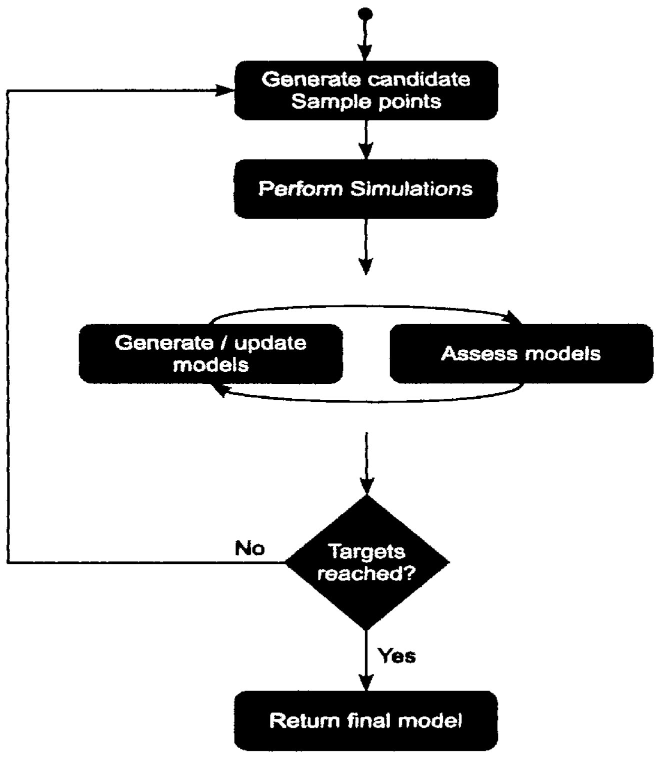


Figure 4.1: General flowchart for adaptive global surrogate modeling

There are two main parts to figure 4.1: an outer sampling loop and an inner modeling loop. Based on a set of simulated data points, the inner loop will generate and improve approximation models until no further accuracy improvements can be made. If the resulting model is acceptable the control flow terminates, else a new set of data points is determined, and the modeling process resumed.

Of course, as-is, figure 4.1 has little practical use since it is very vague. Like the Expectation-Maximization algorithm [232] it is really a meta-algorithm that can have many different concrete instantiations. Unsurprisingly, a simple review of the literature learns that it is indeed this control flow that forms the common denominator across different publications on (global) surrogate modeling. The difference between implementations being that each starts from a specific method (e.g., Kriging) or problem (e.g., electromagnetic modeling) in order to arrive at a concrete, specific instantiation of figure 4.1. For example [22] and [23] essentially employ the same control flow, except highly customized towards Kriging models and Bayesian models respectively. There are, of course, also a number of more integrated, generalized surrogate modeling stacks, most being geared towards SBO. See for example [136, 233] and the overviews given in [2, 4, 44]. Considering the focus on global models, more relevant works are [22, 25], the Efficient Robust Concept Exploration Method (E-CREM) developed by Lin [9], and the Automatic Model Generation method developed by Zhang et. al. [234].

Instead of deriving figure 4.1 from a concrete method or application (bottom-up) the SUMO Toolbox was driven by an explicit top-down approach: identifying the problems, available techniques, applications, and actors in the modeling process and then determining how they can be brought together in a flexible, extensible manner, using the generic control flow in figure 4.1 as a backbone.

4.4 Software Architecture

The design and implementation problem that arises from the design goals listed in section 4.2.2 is far from trivial since they impose contradictory requirements on the software architecture. Trying to merge all scenarios into a monolithic framework is not an option since it would require too many compromises to be useful. Instead we adopt the microkernel design pattern which is commonly associated with operating systems research [235]:

> *The Microkernel architectural pattern applies to software systems that must be able to adapt to changing system requirements. It separates a minimal functional core from extended functionality and customer-specific parts. The microkernel also serves as a socket for plugging in these extensions and coordinating their collaboration.*

The idea is to wrap the core controlling algorithms into a flexible microkernel and moving all extra (possibly problem specific) functionality into separate plugins. The communication with these plugins happens through well defined Application Programmer Interfaces (APIs). If the APIs are defined cleanly, with the right level of abstraction, this allows for a very flexible software architecture that can be molded to fit a wide range of environments. The disadvantage of a microkernel approach, though, is that the modularity adds some extra communication overhead over the case where all algorithms are tightly coupled. In operating systems this overhead can become problematic. However, given the topic of this dissertation we consider this overhead to be negligible, especially compared to the simulation cost. The benefits it incurs (flexibility, customization) greatly outweigh the disadvantages.

Another important motivation for selecting a microkernel-type software architecture is because we wish to maximize code and software reuse. Software reuse is the process of creating software systems from existing software rather than building software systems from scratch. This simple yet powerful vision was introduced in 1968 [236] and the benefits that result from it are obvious [237]:

- reduced effort duplication
- higher quality components (increased dependability)
- effective use of specialists
- accelerated development, resources can be spent on 'the interesting parts'

The key to proper software reuse is the appropriate definition of different levels of abstraction. *"It is well known that it is not always advisable to make everything generic and parameterisable (over-engineering* or *gold-plating) as the costs may outweigh the benefits due to increased complexity. On the other hand we do not only abstract to allow reuse but to facilitate understandability by reducing complexity"* [238]. An important part of the work involved in this dissertation was trying to find a good balance between these trade-offs.

Eventually this resulted in the design of a plugin-based infrastructure for the SUMO Toolbox using standard object oriented design patterns and fully exploiting polymorphism. This will be discussed in detail in section 4.5. An illustration of the plugin-based infrastructure is given in figure 4.2. The figure shows how different plugins can be composed into various configurations or replaced by custom, more problem specific plugins. In this way many different concrete instantiations of the control flow in figure 4.4 can be realized by selecting the appropriate subset of plugins. The plugins currently available include:

- *Data sources:* native executable or shell script, Java class, Matlab script, user defined
- *Model types:* interpolation (linear, cubic,natural neighbor), polynomials, rational functions, Multi Layer Perceptrons, RBF models, RBF Neural Networks, Support Vector Machines (ε-SVM,ν-SVM,LS-SVM), ordinary kriging, blind kriging, GP models, smoothing splines, hybrid (ensembles), user defined
- *Model parameter (hyperparameter) optimization algorithms:* hill climbing, BFGS, Pattern Search, GA, PSO, DIRECT, simulated annealing, NSGA-II, Differential Evolution, EGO, Sequential Quadratic Programming (SQP), user defined

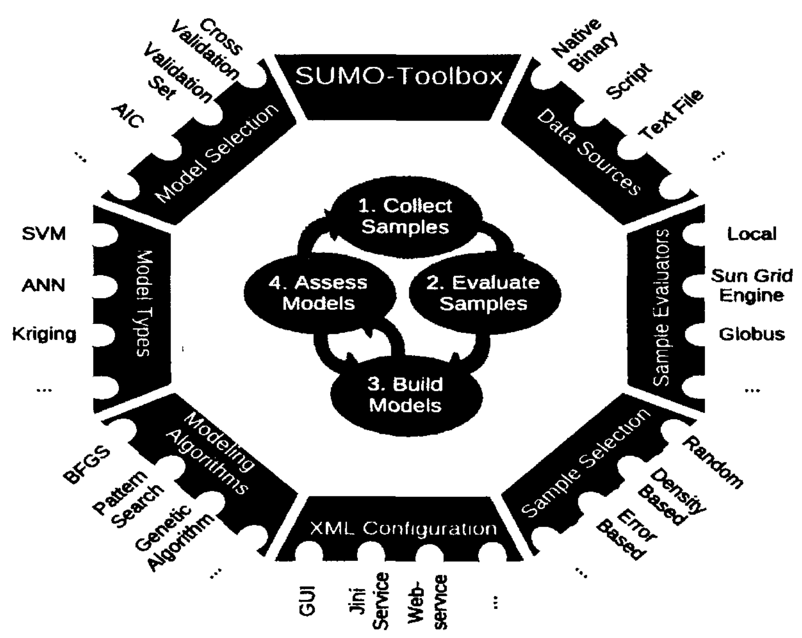


Figure 4.2: SUMO Toolbox Plugins. Based around a small coordinating 'kernel' many plugins are available that can be composed into various configurations.

- *Initial experimental design:* random, Central Composite, Box-Behnken, (Optimal) Latin Hypercube, Voronoi, full factorial, user defined DOE
- *Adaptive sample selection:* error-based, density-based, gradient based, LOLA [239], random, optimization driven infill sampling[1], Mode-Pursuing Sampling (MPS) [240], and various hybrids
- *Model selection:* cross validation, validation set, Leave-one-out, AIC, Linear Reference Model (LRM), model difference, user defined

Some of these plugins are implementations of existing algorithms (e.g., MPS [240]) while others are the result of our own research (e.g., LOLA [239], LRM [241]). Together the different available plugins already cover a very wide spectrum of techniques and problems. If additional, highly problem specific plugins are needed, these can be incorporated in a straightforward way as is described in the following sections.

[1]This is an extended version of the EGO algorithm originally developed by Jones et. al. [89]. It includes extensions discussed by [90] and provides a configurable algorithm, supporting constraints, that balances global sampling (exploration) and optimization driven sampling (exploitation). More details are given in section 4.6.

4.5 Code organization

From a high level point of view the SUMO Toolbox consists of 4 major subsystems, these are illustrated in figure 4.3. The central entity is the control system, it imple-

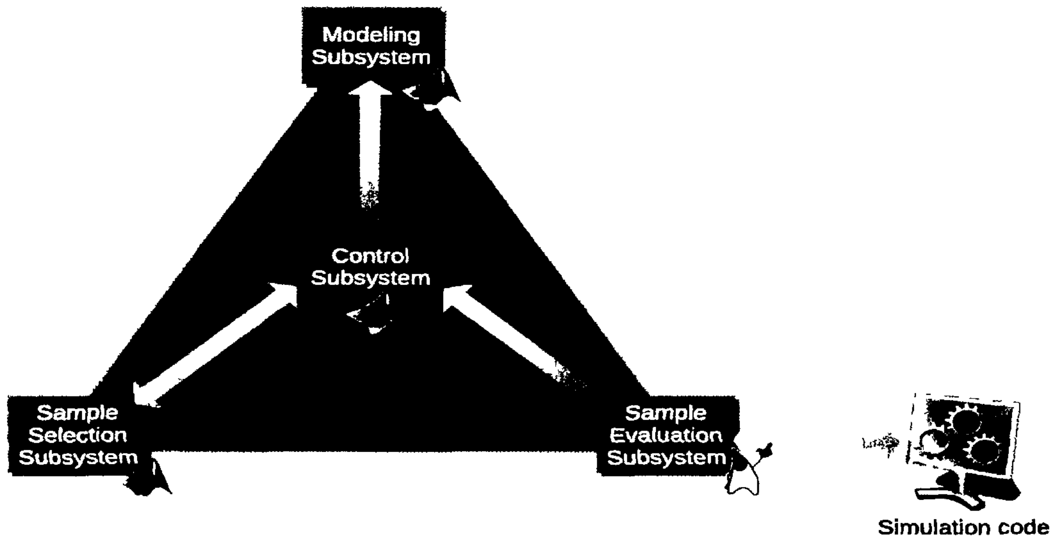


Figure 4.3: The different subsystems that make up the SUMO Toolbox

ments the main control flow and coordinates between the other subsystems. The modeling subsystem is responsible for the generation of approximation models, the sample selection subsystem for the identification of interesting simulation points, and the sample evaluation subsystem is responsible for ensuring the selected points get simulated efficiently. All subsystems are implemented in Matlab, except the sample evaluation subsystem which is implemented in Java (to allow for threading and easy integration with distributed resources). Since Matlab has a Java Virtual Machine built into its framework, interaction between both languages was trivial to achieve.

Matlab was chosen as the base platform for a number of reasons:

- the large number of toolboxes and implementations of standard and advanced numerical routines it provides
- complete and up to date documentation of said toolboxes and routines
- operating system independence
- the seamless integration with Java
- the large surrounding user community

A disadvantage of using Matlab, though, is that the toolbox is locked to a proprietary platform (with associated license fees) over which we have no control. This as opposed to the many free software alternatives which provide more freedom, flexibility, and security in the long term. However, the advantages listed above, and given that Matlab is the de facto standard platform for scientific computing tipped the decision in favor of using Matlab.

The remaining subsections of this section will discuss each major subsystem in detail as well as a number of other smaller, supporting subsystems. This part of the thesis is the most prone to becoming obsolete. Therefore we shall concentrate on the high level structure of the different components, rather than focus on the implementation and purpose of every single class. Thus the class- and sequence diagrams given in the remainder of this section should not be taken as a perfect bijective mapping of the underlying source code. Some details and utility classes have been omitted for the sake of clarity.

4.5.1 Control subsystem

We start with the components that are responsible for bootstrapping the toolbox and executing the control flow. Together they form the control subsystem that drives the execution of the toolbox. It is instructive to go over the main flow of events as illustrated in figure 4.4.

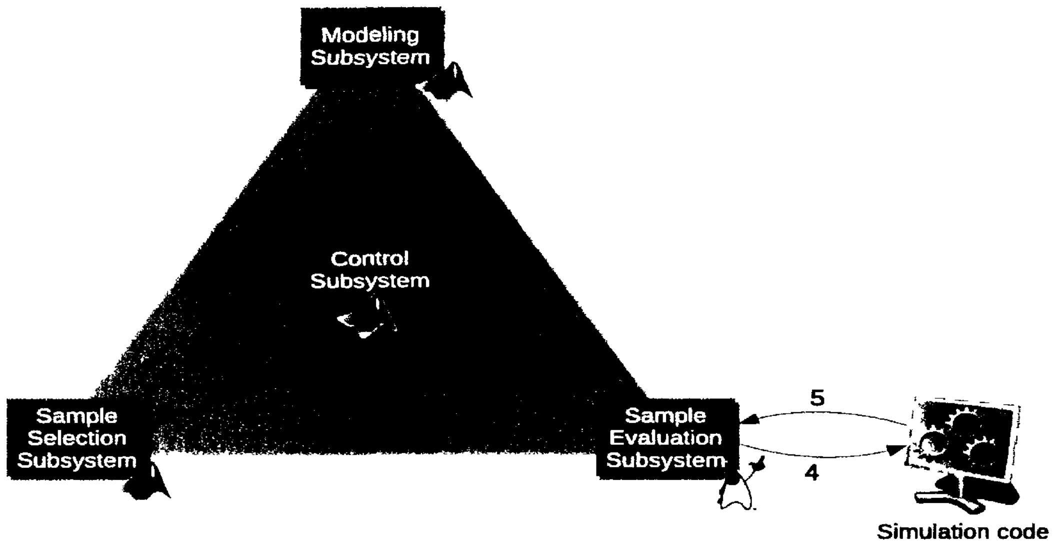


Figure 4.4: SUMO Toolbox Control Flow

After some initialization and bootstrapping, the control subsystem requests the sample selection subsystem to generate an initial DoE in step (1). The DoE is generated and returned to the control code in step (2) which then schedules it for evaluation in step (3). Evaluation of the simulator is handled by the sample evaluation subsystem which returns the evaluated points to the control code when finished (steps 4 to 6). Data is now available to start the modeling so the control code instructs the modeling code to start generating models for the given points (step 7). Once the modelbuilding process has been completed, control passes back to the control subsystem (step 8) which decides if the results are good enough to terminate the model building process. If not, the sample selection subsystem is again triggered and the whole process starts again.

This description of events is already more concrete than the flowchart in figure 4.1 but still does not refer to the low level classes that are actually involved in the implementation. That is illustrated by the sequence diagram in figure 4.5 which works as follows:

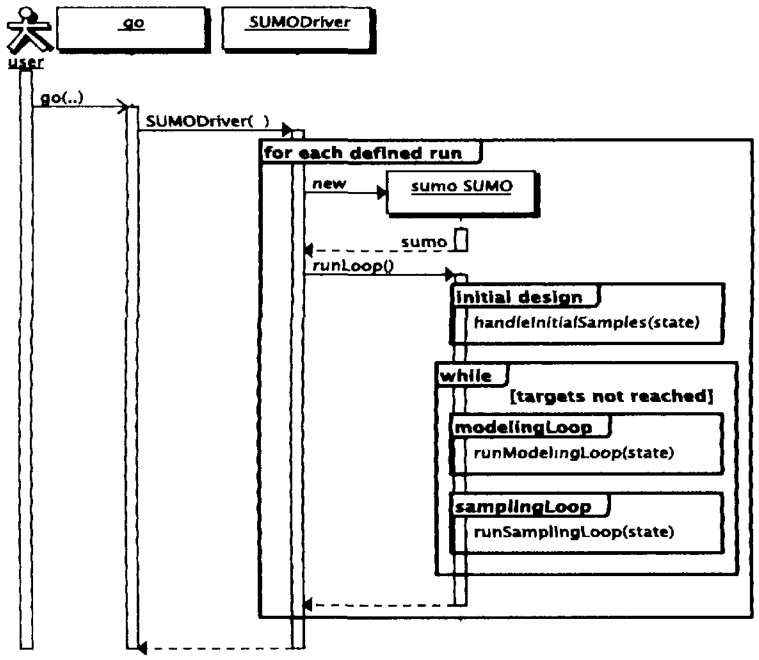


Figure 4.5: Sequence diagram of the bootstrapping and control subsystem

The user starts the toolbox by executing the Matlab *go* script. This script performs some basic sanity checking (command line arguments, is Java enabled, is the Matlab version supported, etc.) and then calls the *SUMODriver* script with the name of the XML toolbox configuration file (cfr. section 4.5.5). The task of *SUMODriver* is then to parse the XML file and execute each *Run* defined therein.

A *Run* represents one surrogate modeling experiment, i.e., the generation of a model for a particular problem with a particular configuration. The toolbox configuration file may contain multiple runs which are executed sequentially. This is useful for setting up benchmarking tests. Each *Run* may freely define the problem to model, and components to use in the model building process.

Thus, for each run, the *SUMODriver* script takes care of reading the XML file, splitting it into pieces and ensuring all necessary objects are instantiated with the relevant XML fragment that contains the configuration options for each object. The most important of these objects is an instance of the *SUMO* class. The SUMO class forms the core of the SUMO Toolbox. Once instantiated, its *runLoop* method is triggered by *SUMODriver*, which then starts the familiar control flow from figures 4.1 and 4.4.

We now consider each of the subsystems involved in this control flow in turn. Starting with the sample selection subsystem.

4.5.2 Sample selection subsystem

The sample selection subsystem encompasses two parts: the *InitialDesign* class hierarchy (for static DoE) and the *SampleSelector* hierarchy (for sequential designs). We discuss each of these in turn.

4.5.2.1 Initial Designs

As the name implies the initial designs are responsible for generating the initial DoE that is needed to seed the sequential design strategy (X_0 from section 3.2). The generation and evaluation of the initial design is handled by the *SUMO* class in its *handleInitialSamples()* method. The implementation hierarchy is very simple (see figure 4.6). There is a single base class *InitialDesign* with one abstract method *[samples] = generate()*. Subclasses should override this method and return a set of points in the input space that should be used as the initial experimental design.

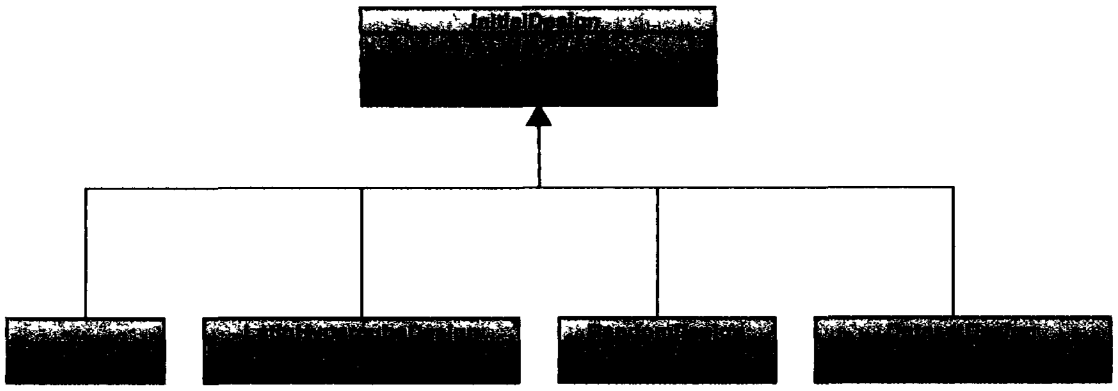


Figure 4.6: Initial Design hierarchy

Of the different subclasses, the most important is probably *LatinHypercubeDesign*. It implements an optimization algorithm by Ye et. al. [242] to generate points on Latin hypercube. Unfortunately, generating a good Latin hypercube design is a very difficult and computationally expensive task with the results often being quite poor [242–244]. For this reason the implementation in the toolbox will first attempt to download a known optimal Latin hypercube design from the internet[2]. Only if a design with the requested dimensionality and number of points is not found will the toolbox fallback to generating one itself.

[2]`http://www.spacefillingdesigns.nl`

4.5.2.2 Sequential Designs

Once an initial set of data points is available it is the task of the sample selection algorithm to incrementally extend this set with maximally informative new data points. The key question of course is, what is maximally informative? As already discussed in section 3.6.2 many diverse criteria can be used to decide whether a particular location in the input space is worth sampling or not. Given this diversity and the natural requirement to use different criteria in concert, possibly even dynamically, this results in a complex set of use cases.

The main abstraction is the *SampleSelector* class, it contains one method which subclasses can override: *[newSamples, priorities] = selectSamples(state).* This is illustrated in figure *4.7.* Given a state with some contextual configuration (problem

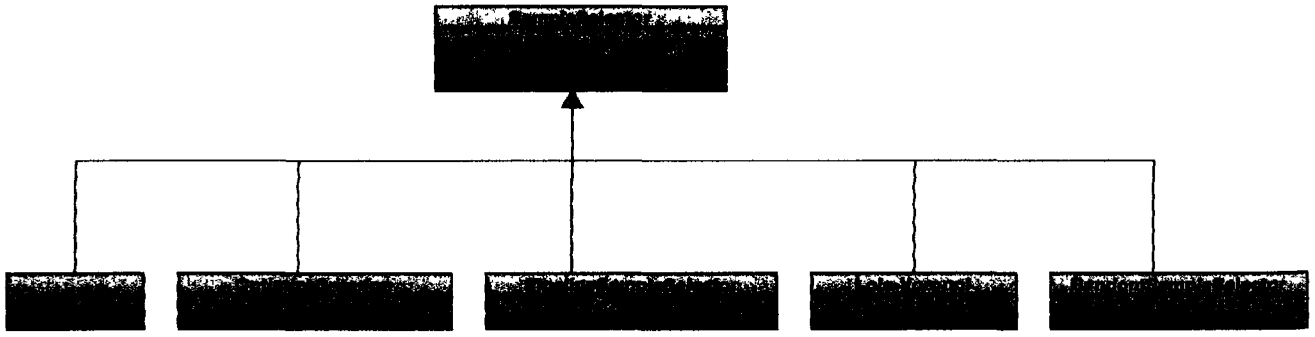


Figure 4.7: Sample Selection hierarchy

dimensionality, best models built so far, number of points to return, etc.) the implementing method should return a new set of potential sample locations. Associated with the list of points is a corresponding list of priorities. The higher the priority the more important the sample location is deemed (see chapter 6). Subclassing *SampleSelector* gives the most possible freedom concerning the algorithm used. However, it also involves the most work. Thus two noteworthy subclasses are defined that make the task of implementing a new sampling algorithm somewhat easier and facilitate code reuse.

The first is the *OptimizeCriterion* sample selection subclass whose structure is shown in figure 4.8. It essentially combines a generic optimizer with a *CandidateRanker* object. The latter is a criterion that returns a score given one or more candidate sample locations. Thus the optimizer simply optimizes the given criterion and returns the found optima as a new sample point location. There is one caveat though. Experience showed that the same sample point[3] could be returned multiple times across

[3]Deciding whether two points in a continuous domain are the same is actually a tricky problem. Exact equality can be checked but intuitively a user would want to specify a tolerance within which points are considered identical (similar to the ε-tube used in SVM theory). However, while such a spherical range can be specified relatively intuitively in 1D and 2D. For more than 2 inputs this becomes difficult to impossible to understand and thus essentially useless. One way to solve this is by specifying the tolerance in relative terms (e.g., relative to the domain which is known to be fixed to [-1 1]). An easier solution, and the one

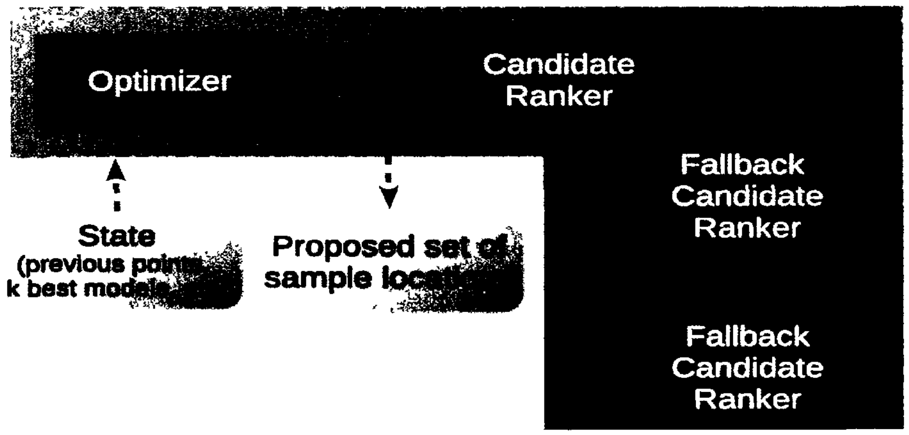

Optimizer
Candidate Ranker
Fallback Candidate Ranker
Fallback Candidate Ranker
State (previous points, k best models)
Proposed set of sample locations

Figure 4.8: The OptimizeCriterion *class. An optimizer drives a candidate ranker in order to find a promising sample location. One or more fallback criteria may be present to ensure the returned points are always unique.*

different sample selection iterations. This happens if the optimizer finds the same optimum multiple times and results in a waste of resources (duplicate points are discarded). Thus *OptimizeCriterion* can be configured with one or more fallback *CandidateRanker* objects. If the first candidate ranker is unable to identify any new unique sample points, the next fallback criterion is triggered and so on. Ultimately, if none of the *CandidateRanker* objects find any new samples, new samples are selected randomly. Alternatively one could add mathematical constraints to the optimizer to ensure it does not suggest points close, or idenitcal to previous points (e.g., as done in [95]).

Sometimes the optimization approach can be slow or have difficulty identifying good sample locations. There are cases were a more direct one-shot approach is easier, faster, and can lead to better sample distributions. This is where the second class, *PipelineSampleSelector*, comes in (figure (4.9)).

Besides the already discussed *CandidateRanker* class, there are two others involved: *CandidateGenerator* and *MergeCriterion*. The candidate generator is responsible for generating candidate sample locations based on some (configurable) one-shot distribution (e.g., a dense grid or voronoi distribution). These potential samples are then passed to the candidate ranker components. Each of these assesses the candidate points according to a criterion (e.g., space filling-ness, response value, model uncertainty, expected improvement, etc.). This results in each candidate point receiving a score (higher is better) from each candidate ranker. These scores are then merged by the merge criterion in order to come to a final distribution of data points. Different implementations of the merging algorithm are possible. Merging may be done based on distance (i.e., space filling), on score, on a hybrid distance-score combination, or

the SUMO Toolbox uses, is to use the manhattan distance instead of the eucliean distance. This effectively results in a hypercube instead of a sphere. One can then specify the tolerance in function of the minimum of the maximum distance in each dimension.

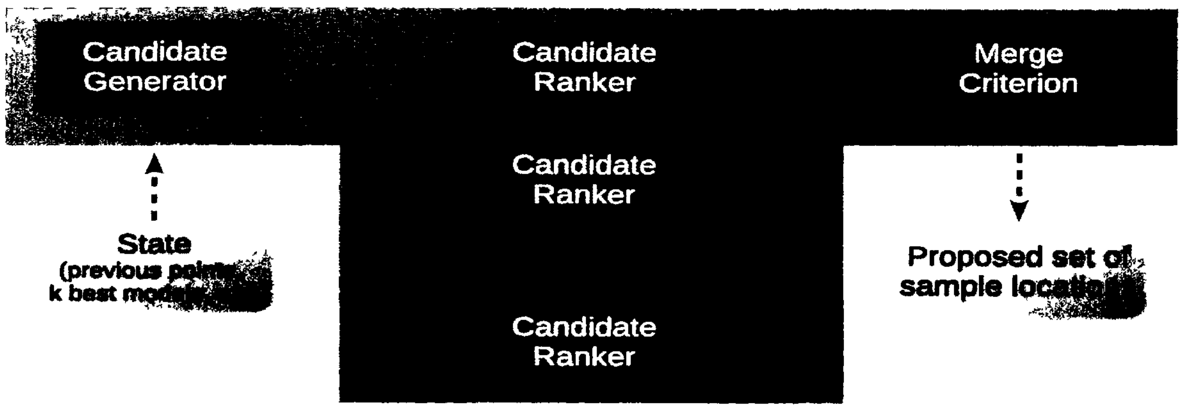


Figure 4.9: The PipelineSampleSelector *class. The candidate generator generates a set of candidate points, these are then ranked by one or more candidate rankers, and then filtered by the candidate merger which returns the final set of points.*

based on some other user defined aggregation function. The merging function may even be dynamic, giving more weight to a particular candidate ranker depending on the modeling progress. For example, initially it is more important to globally map the design space, while as the model quality improves, the sampling strategy can shift to focusing on the interesting, local features of the domain.

The *PipelineSampleSelector* and *OptimizeCriterion* sample selector classes require each criterion to act on exactly the same set of candidates. A criterion is not allowed to add new candidates. This makes it possible to combine the scores of multiple candidate rankers in a meaningful way. However, this has as a disadvantage that the domain considered by each candidate ranker is the same. This domain is determined by the candidate generator and optimizer which is shared by all ranker objects. Instead it may be interesting to use different optimizers or different candidate generators together. For this there is the *CombinedSampleSelector* class.

The *CombinedSampleSelector* allows one to use different *SampleSelector* objects (not candidate rankers!) together (figure 4.10). For example, it allows you to use an *OptimizeCriterion* object that samples the model optima, together with a pipeline sample selector object that ensures a space filling coverage. Again a *MergeCriterion* object is used to merge the samples proposed by the different algorithms in an intelligent and/or dynamic way.

4.5.2.3 Extensions

There three extensions to the sequential design system outlined so far that the toolbox supports. Firstly, the sample selection infrastructure allows for auto-sampled inputs. Sometimes a particular input is sampled automatically by the simulator. For example, in an electronics context the parameter space may consist of 3 parameters: 2 geometrical design parameters and the frequency. If the simulator simulating this problem uses

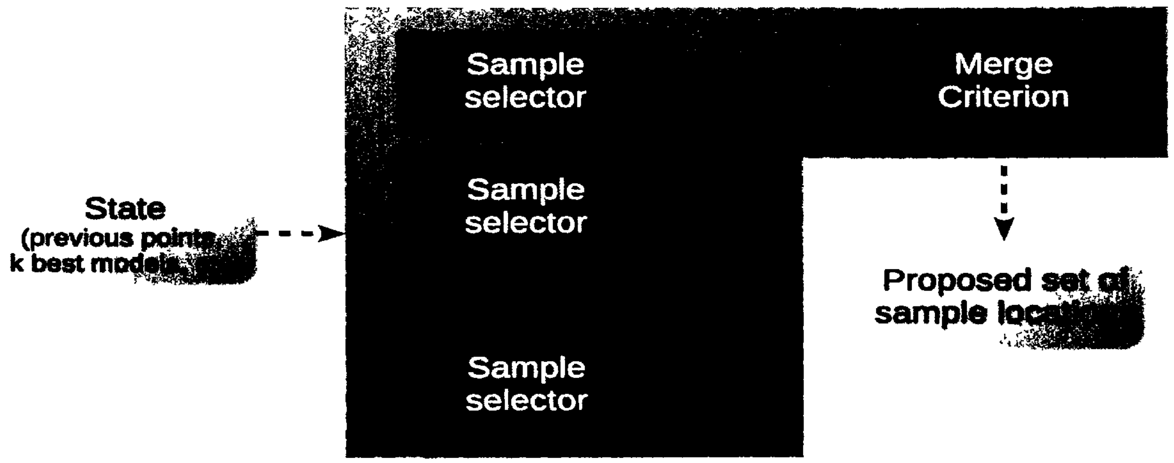

Sample selector
Merge Criterion
State (previous points, k best models)
Sample selector
Proposed set of sample locations
Sample selector

Figure 4.10: The CombinedSampleSelector *class. Multiple sample selectors can be used together. A MergeCriterion object is used to merge the different sets of samples intelligently.*

a frequency based solver, the simulation engine will be able to (relatively) cheaply generate a full sweep of the frequency response for a given value of the geometric parameters. Essentially this means that the model should be constructed with 3 parameters but that sampling should only take into account 2 parameters since the frequency is sampled by the simulation engine (auto-sampled).

Secondly, a sample selection algorithm can be applied on multiple outputs simultaneously, taking into account their correlation. So if multiple outputs need to be modeled together (see for example the applications in [245, 246]), the sample selection algorithm can take this into account in order to propose new sample locations.

Thirdly, in most cases the input domain under consideration is rectangular and defined by simple box constraints. However, it is not unusual for the domain to contain infeasible regions, delineated by linear or nonlinear constraints. See for example the work in [95]. If these are known in advance the different sample selection algorithms can take them into account. In the *OptimizeCriterion* system the level of constraint support depends fully on the optimizer used. In the pipeline system constraint support depends on the candidate generator and/or merge criterion used.

4.5.3 Sample evaluation subsystem

Once promising sample points have been selected it is the task of the sample evaluation subsystem to ensure that the necessary simulations are performed. There are two main parts to this subsystem: the *SampleEvaluator* hierarchy and the *SampleQueueManager* hierarchy. The sequence of events showing how they interact is illustrated in figure 4.11. This is an important figure and should be kept in mind when going through the remainder of this subsection.

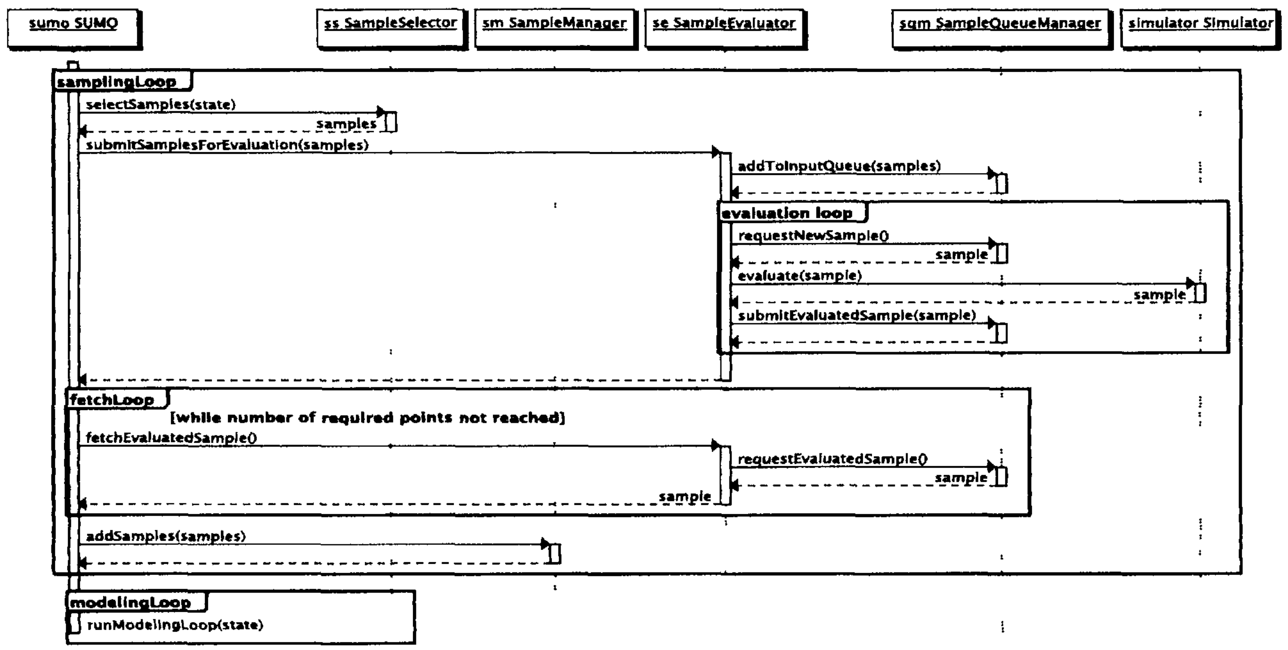


Figure 4.11: Sequence diagram showing how the different components in the sample evaluation subsystem work together.

4.5.3.1 *SampleEvaluator* hierarchy

The central class in the sample evaluation subsystem is the *SampleEvaluator* interface. It defines methods for submitting *SamplePoint* objects for evaluation, and retrieving them once they have been evaluated. Depending on which *SampleEvaluator* implementation is used, evaluation (calculating the response output for a given input location) can occur through different means: a Matlab script, a native executable, a data set, a Java class, etc. The class hierarchy is shown in figure 4.12.

The *SampleEvaluator* usually runs in its own thread (if one of the threaded subclasses are used). For example, the *LocalSampleEvaluator* employs a pool of threads to perform simulation evaluations. This allows it to take advantage of multi-core or multi-CPU hardware. More importantly, it allows sample evaluation (i.e., simulations) to run in parallel, without blocking the modeling code (native Matlab only supports a single thread of control). The result of this is that communication between the control code and the sample evaluation code occurs asynchronously through the use of queues. Managing these queues is the task of the sample queue manager component.

4.5.3.2 *SampleQueueManager* hierarchy

The Sample Queue Manager (SQM) is responsible for managing the input and output queues that facilitate the communication between the control subsystem and the sample evaluator (see figure 4.13). Points requested by the sample selection algorithm are

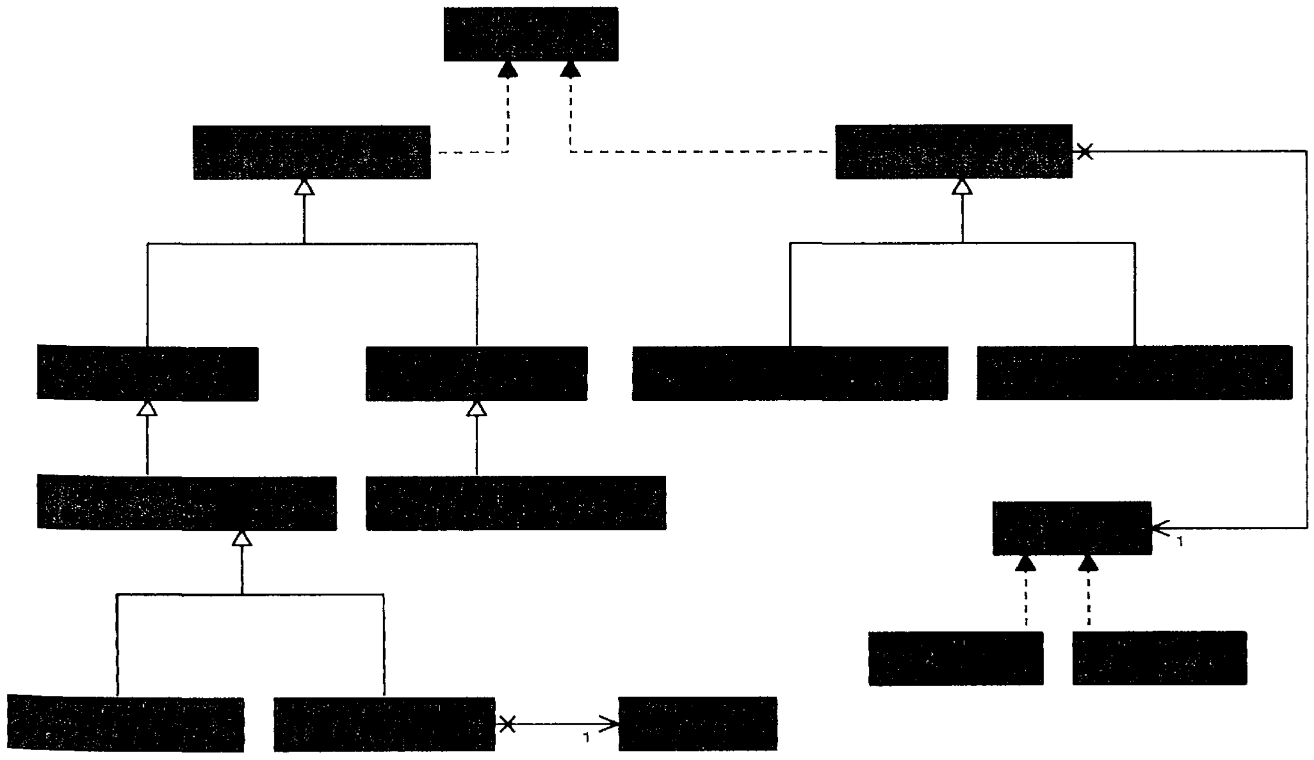


Figure 4.12: Sample evaluation hierarchy

placed on the input queue which, in turn, is emptied by the sample evaluator. Once a point has been evaluated, the sample evaluator places it on the output queue where it can be retrieved again by the control code.

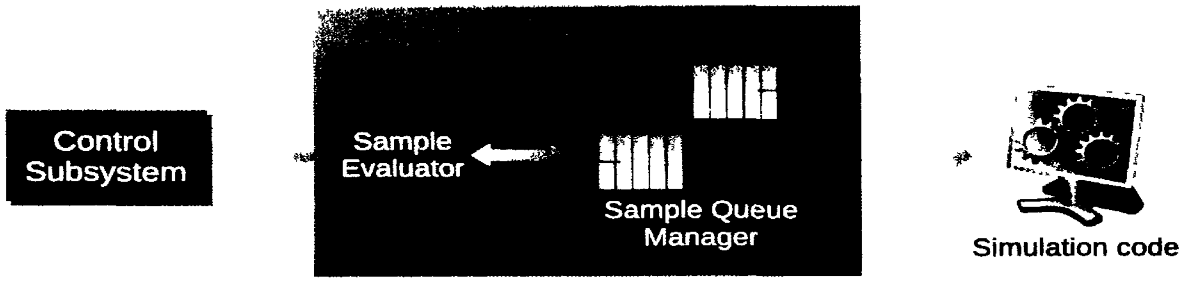


Figure 4.13: The sample queue manager forms the bridge between the control subsystem and the sample evaluation backend.

Besides input and output queue's, the SQM also keeps track of all sample points that are currently being evaluated. To this end a separate thread (*PendingMonitorThread*) is running which continuously monitors how long each point has been simulating, and what the average simulation time is per point (taken over a sliding window). The user is then able to specify a timeout. If the simulation time of a point exceeds k times the average simulation time, the point is considered lost and discarded from the pending set. This mechanism prevents the toolbox from waiting on points where the simulation engine is failing to converge. If the point does eventually come back successfully (i.e., not failed) it is still included in the modeling, albeit with a warning.

The input queue of the SQM is itself pluggable, allowing for different queue management systems (see figure 4.14). This can be particularly useful in a distributed context (cfr. chapter 6). By default two simple implementations are available: *BasicSampleQueueMangager*, and *PrioritySampleQueueManager*. The first implements a simple FIFO queue, while the second implements a priority queue where samples with the highest priority (as determined by the sample selector) are scheduled first.

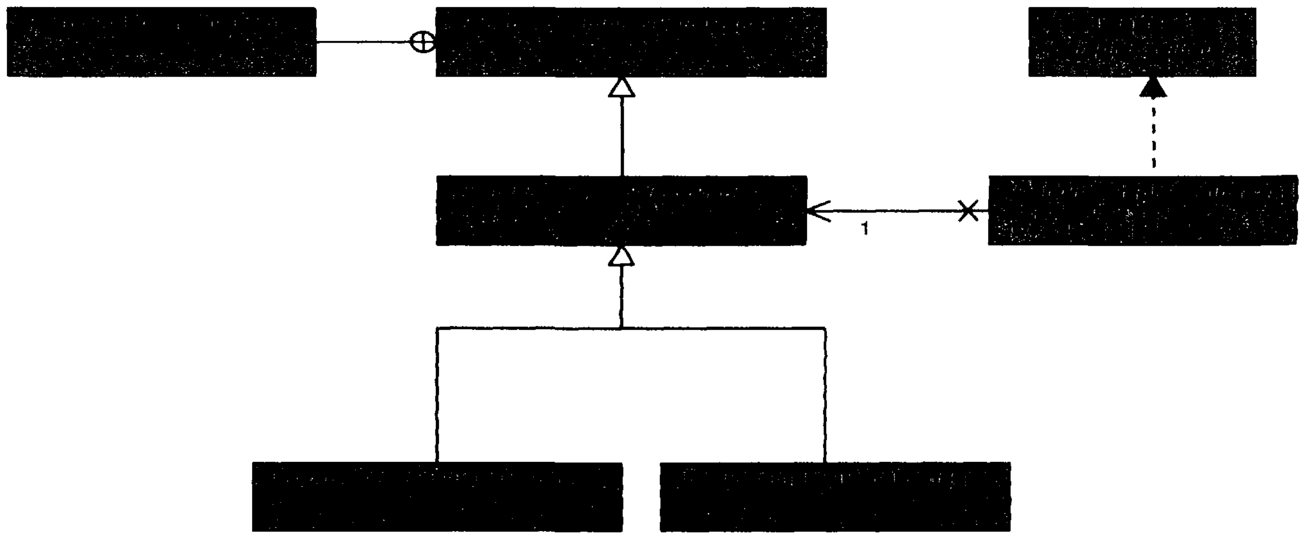


Figure 4.14: Sample queue manager hierarchy

4.5.3.3 Data post-processing

Sometimes it is necessary to post process data, for example to take the logarithm of the response value if it covers a large range. To prevent from having to change the simulator implementation itself, the SUMO Toolbox provides a *DataModifier* base class (figure 4.15). Subclasses override its only method *[out] = modify(in)* which modifies the given response values in some way. For example, this may be done to improve the modeling process (e.g, multiply by some scaling factor) or to stress test the modeling code by applying a modifier that adds outliers or noise to the response data.

Calling the *modify(..)* method is done in the *SampleManager* class. Besides applying modifiers, this class takes care of selecting the appropriate inputs/outputs, removing out of range samples, scaling points to/from model space and other similar actions.

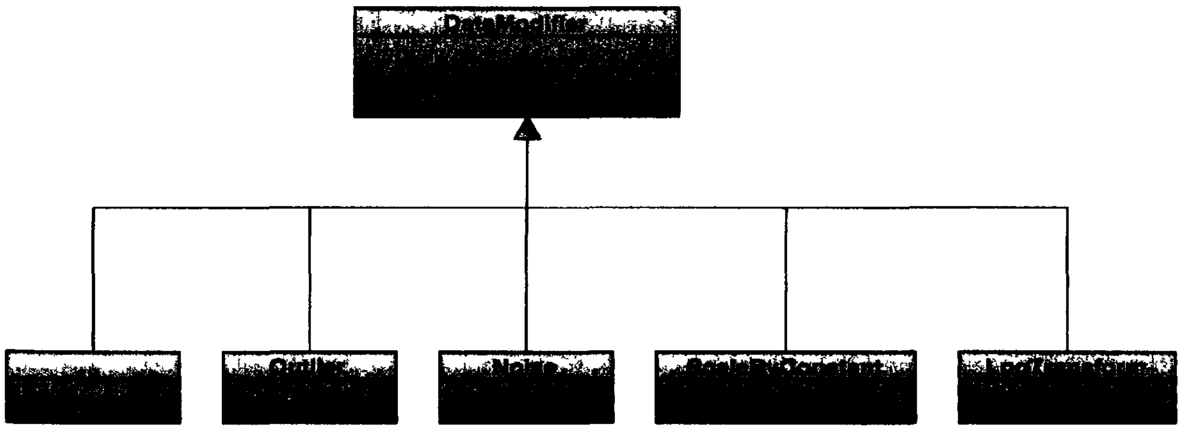


Figure 4.15: Data modifier hierarchy

4.5.4 Model generation subsystem

The fourth major subsystem of the toolbox is the one responsible for generating and fitting models. The main concepts involved are: model type, model factory, model builder, and measure. How these relate to the model fitting, model parameter optimization, and model selection problems is shown in the sequence diagram in figure 4.16 and explained in detail in the following subsections. Again it is important to refer to figure 4.16 throughout the discussion below.

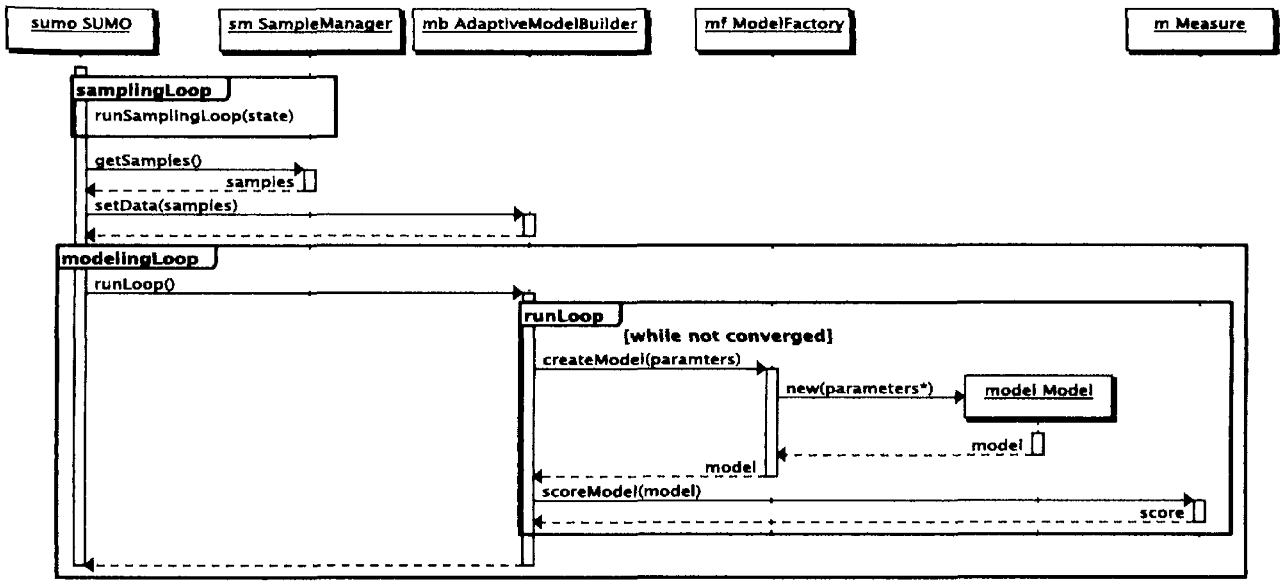


*Figure 4.16: Sequence diagram showing how the different classes in the model building subsystem inte*ı*act.*

4.5.4.1 Model types

Arguably the most important class in the SUMO framework is the *Model* class. *Model* serves as the base class for every supported surrogate model type: neural networks, support vector machines, Kriging, and many others. Thus the *Model* class encapsulates the essence of a function approximation technique that is able to generalize from given data. Methods include:

- *construct(samples, values)* : fit a model on the given data
- *evaluate(samples)* : evaluate the model at the given data points
- *plotModel(..)* : plot the model
- *getDimensions()* : get the number of inputs and outputs
- *complexity()* : return the complexity of the model as a scalar value (e.g., number of parameters)
- *getDescription()* : return a user friendly description of the model as a string.

Many more methods are available, but these are the most important. Some methods of *Model* have the suffix *InModelSpace* in their name. To the outside world the toolbox always expects data to be in *simulator space*, i.e., in the original domain determined by the problem. However, internally the toolbox works exclusively in the domain $[-1, 1]^d$. This is referred to as model space. Thus a method that deals with data typically forwards the call to to variant with the *InModelSpace* suffix after scaling the data to model space. For example, an *evaluate(..)* call will arrive at the base class implementation which performs the scaling and forwards the call to *evaluateInModelSpace(..)* which performs the actual operation.

There is only one abstract method that a subclass *must* implement:

- *evaluateInModelSpace(samples)* : evaluate the model at the given set of points, where the points are defined in model space.

Remark that the *constructInModelSpace(..)* method is not abstract. This is because not all models have a fitting phase. Take for example models based on lazy learning (e.g., nearest neighbour, locally weighted regression).

Besides the two abstract methods a subclass may override other methods with more efficient, model specific implementations. Two good examples in this respect are:

- *getExpression()* : return a closed form expression of the model. This method makes it possible to export a model object to a SUMO Toolobx independent format.
- *evaluateDerivative(samples)* : evaluate the model derivative at the given points (the base implementation uses a numeric approximation)

For a full list of methods and subclasses the reader is referred to the implementation.

As an aside, the model subclasses do not have to be implementations of pure approximation techniques. Subclasses may be written that wrap existing models in order to change their behavior or hide particular inputs or outputs. For example the *ComplexWrapper* class wraps a model with two outputs (representing the real/imaginary part of the data) in a new model that has one complex valued output. This model can then be used transparently just as any other. Other model types in this spirit include:

- *EnsembleModel*: wraps one or more models in a weighted ensemble
- *ExpressionModel*: wraps any analytical expression as a SUMO Model object
- *DataModel*: wraps a fixed dataset in a model object. This is useful to explore a high dimensional data set using the GUI model browser in SUMO.
- *OutputFilterWrapper*: a wrapper that can transparently add or remove outputs from a model

Many other (hybrid) scenarios are possible. For example one could create a new model class that implements a space mapping or difference mapping model.

Once a model has been generated, the toolbox provides a GUI to load, expore, and asses the quality of the generated model. A screenshot of the model browser and its model information window is shown in figures 4.17 and 4.18.

4.5.4.2 Model builders

At the other end of the model generation spectrum we encounter the model builder classes. Recall from section 3.7.2 that each model type is characterized by a set of hyperparameters θ that control the structure and complexity of the model. The optimal choice of θ can be determined through an optimization process (cfr equation 3.2). In the toolbox this optimization process is coordinated by the *AdaptiveModelBuilder*. Depending on the subclass, a different optimization algorithm is used to optimize the model parameters (see figure 4.19).

The *AdaptiveModelBuilder* class encapsulates the common logic between different model builders such as: managing and evaluating the different measures (cfr. section 4.5.4.4), tracking the modeling progress through a number of profilers (cfr. section 4.5.7), saving intermediate results, managing the best model trace (cfr. section 4.5.4.5), and implementing the different restart strategies (cfr. section 4.5.4.6). In this way the different subclasses can focus on the optimization part itself.

The main method from *AdaptiveModelBuilder* that drives the model generation is the *runLoop(..)* method. The base class implementation generates a single, fixed model by invoking the *createModel()* method from the nested *ModelFactory* class (see section 4.5.4.3). Thus in the base implementation there is no optimization, the model parameters are fixed or can be optimized by the model type itself (for example Kriging

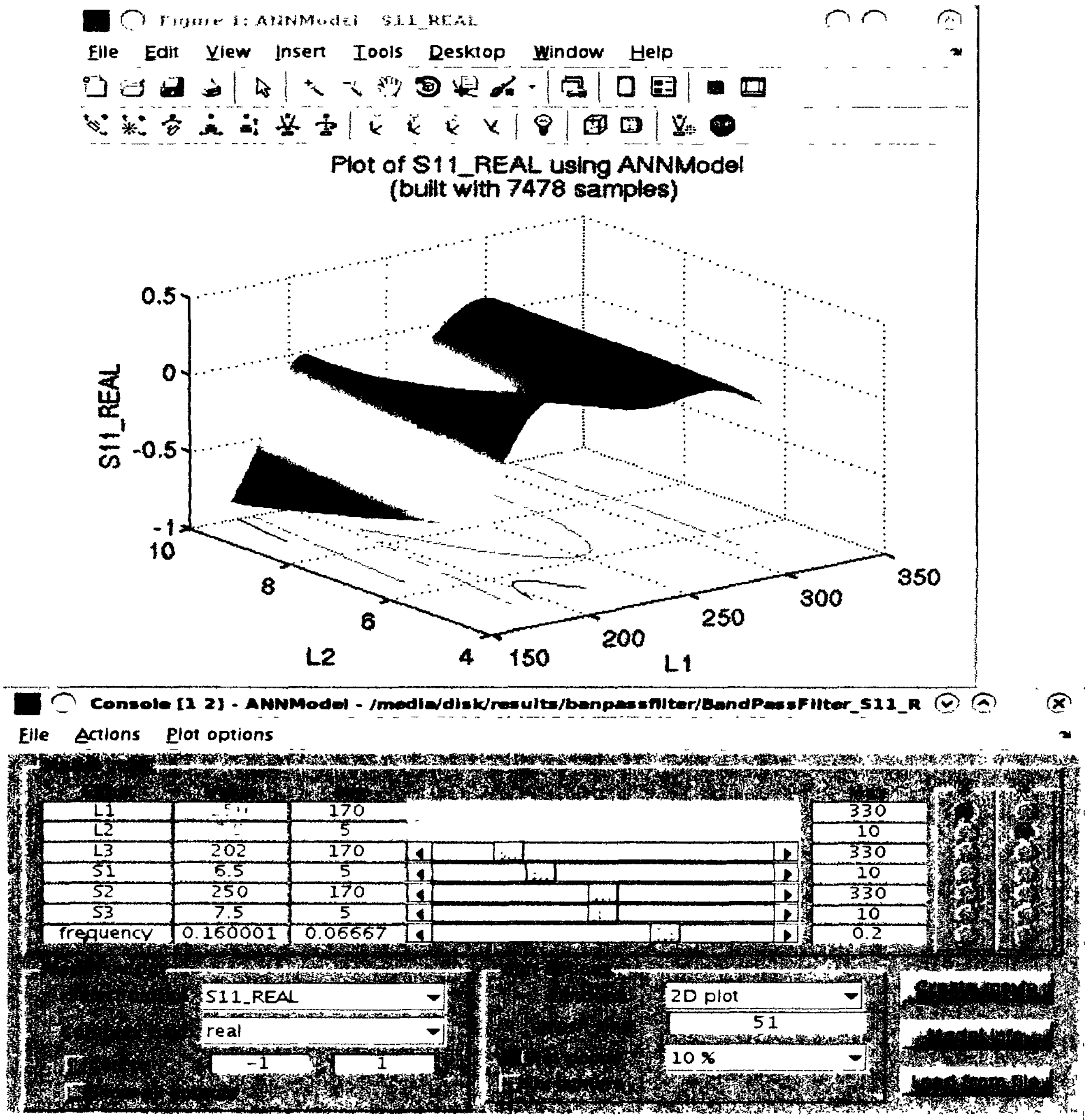


Figure 1: ANNModel S11_REAL
File Edit View Insert Tools Desktop Window Help
Plot of S11_REAL using ANNModel
(built with 7478 samples)
S11_REAL
0.5
0
-0.5
-1
10
8
6
4
L2
150
200
250
300
350
L1
Console [1 2] - ANNModel - /media/disk/results/banpassfilter/BandPassFilter_S11_R
File Actions Plot options
S11_REAL
real
2D plot
10 %

L1	[illegible]	170	330
L2	[illegible]	5	10
L3	202	170	330
S1	6.5	5	10
S2	250	170	330
S3	7.5	5	10
frequency	0.160001	0.06667	0.2

-1 1

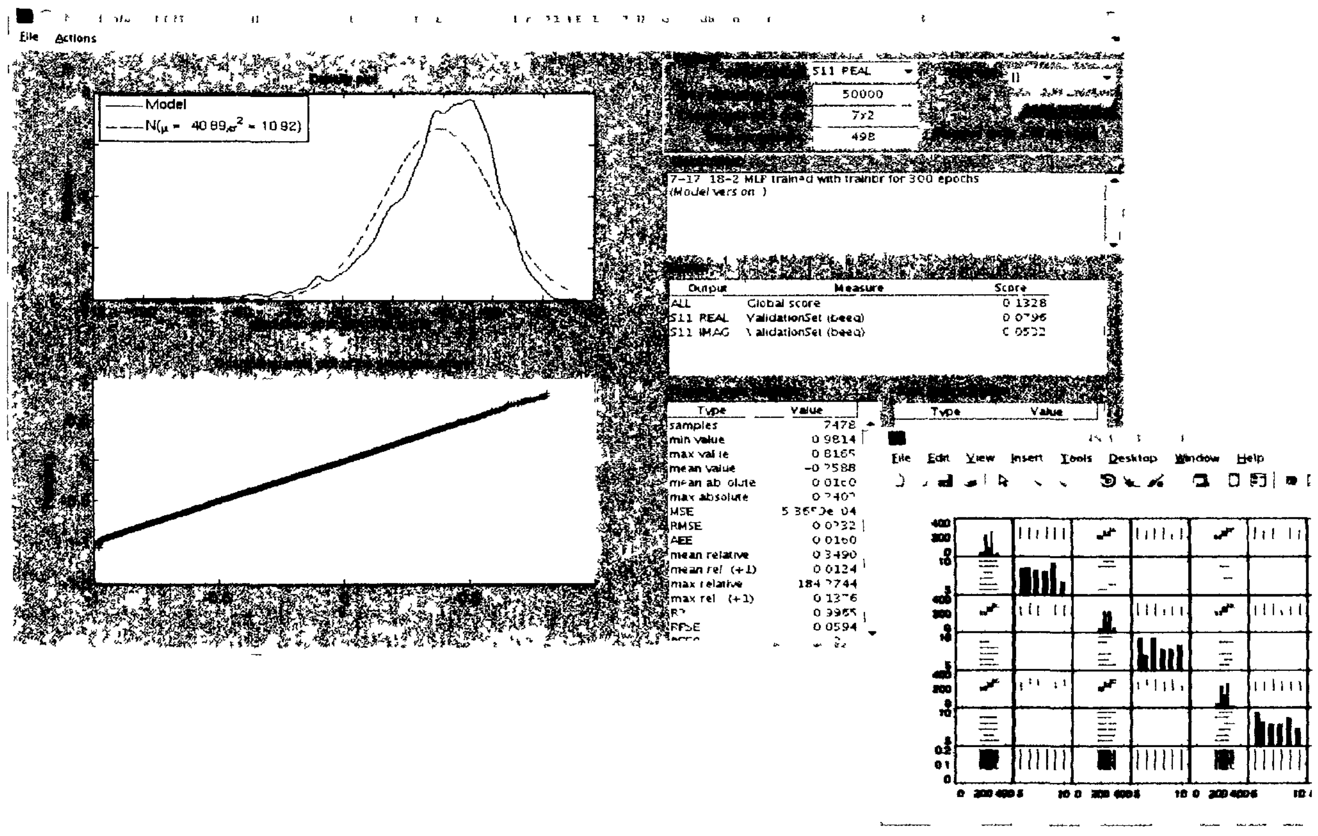
51

Figure 4.17: A screenshot of the model browser GUI

models often implement their own specific hyperparameter optimization method based on the likelihood).

The main model builder subclasses in the SUMO Toolbox are:

- *OptimizerModelBuilder* : wraps any of the optimizers from the SUMO Opti-



S11 PEAL
50000
7x2
498

7-17 18-2 MLP trained with trainbr for 300 epochs
(Model vers on)

Output	Measure	Score
ALL	Global score	0 1328
S11 REAL	ValidationSet (beeq)	0 0796
S11 IMAG	ValidationSet (beeq)	C 0532

Type	Value
samples	7478
min value	0 9814
max value	0 8165
mean value	-0 7588
mean absolute	0 0160
max absolute	0 2407
MSE	5 3650e 04
RMSE	0 0232
AEE	0 0160
mean relative	0 3490
mean rel (+1)	0 0124
max relative	184 2744
max rel (+1)	0 1376
R²	0 9965
RRSE	0 0594

Type	Value

Figure 4.18: A screenshot of the model info GUI and sample distribution plot. Both of which can be opened from the model browser in figure 4.17.

mizer hierarchy (cfr. section 4.5.6). Thus this class provides a very fast and easy way to try out different optimization algorithms for optimizing the model parameters. All that is needed is an *Optimizer* subclass.

- *SequentialModelBuilder* : implements a kind of evolutionary strategy. *SequentialModelBuilder* maintains a sliding window history of previously constructed models and sequentially asks the model factory to generate a new model based on the history. Effectively this class allows for a fully custom model parameter optimization method.

- *GeneticModelBuilder* : model parameter optimization is done based on a GA. The difference with the *OptimizerModelBuilder* class is that *GeneticModelBuilder* supports a population of model objects with model type specific operators (versus a population of real valued vectors and classic genetic operators).

- *ParetoModelBuilder* : a base class for multi-objective hyperparameter optimization (see chapter 8)

- *EGOModelBuilder* : a model builder that implements the EGO algorithm from

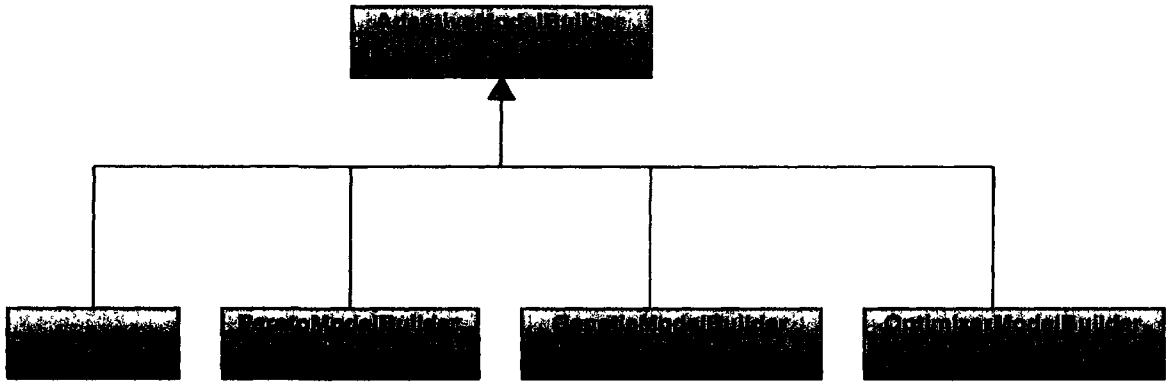


Figure 4.19: Model builder hierarchy

Jones et al. [89]. Internally a blind Kriging model is used to predict which areas of the hyperparameter optimization landscape should be sampled to achieve better quality models.

Together these subclasses already cover a very wide range of strategies for optimizing the model parameters. Others, e.g., one based on Bayesian inference, can easily be added.

4.5.4.3 Model factories

The *Model* class hierarchy encapsulates the features of a particular fitting method while the *AdaptiveModelBuilder* class encapsulates the algorithm for setting the model parameters. However, there is still one piece missing from the puzzle. While the implementation details of each fitting type are hidden behind the Model interface, constructing an object of a particular model type still requires knowledge of the specific parameters involved. This means that, as-is, a model builder class would need to be aware of the specific, different ways of instantiating each model subclass. Something that should be avoided from a software engineering viewpoint. This is where the model factory class hierarchy comes in, an implementation of the factory method design pattern [247] shown in figure 4.20.

A model factory object sits in between the model type and the model builder. It abstracts away the instantiation and configuration of individual model types so that a model builder can operate on many different model types without having to change any code. The model builder does not interact with the model objects directly, it only interacts with the model factory. The sequence of events is illustrated in figure 4.16. A model builder instance instantiates a model factory class with the relevant XML configuration fragment and repeatedly queries it for model objects in its *runLoop* method (which implements some optimization algorithm).

The goal of the factory pattern is to abstract away the concrete instantiation details of individual model types from the model builder through the use of a factory

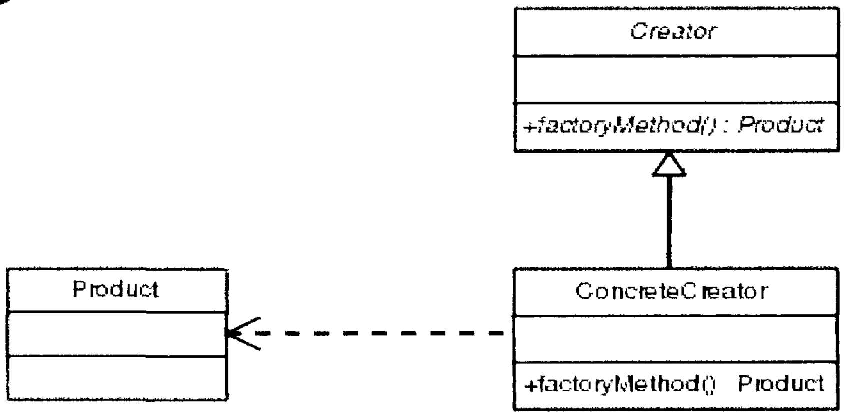


Figure 4.20: The factory method design pattern

method. However, different model builders may require slightly different factory methods. Thus, depending on which model builders should be supported by a model factory for a specific model type, the set of implemented factory methods will differ. For this reason the base model factory class, *ModelFactory*, can have a number of subclasses (see figure 4.21).

An important subclass in this respect is *GeneticFactory*. *GeneticFactory* requires subclasses to override crossover and mutation methods in addition to the pure creational factory methods from *ModelFactory*. The two required factory methods from *ModelFactory* (*createModel(..)*, and *createInitialModels(..)*) are usually sufficient for most model builders.

Thus, in sum, adding support for a new model type in the toolbox reduces to adding a new Model subclass and corresponding *ModelFactory* subclass. Which factory methods should be implemented will depend on the *ModelFactory* class that is used as a base. But again, subclassing *ModelFactory* is usually enough. A concrete example is given in section 4.8.

4.5.4.4 Model selection

The whole model building process is of course useless without some performance metric to guide the model parameter optimization. Recall from section 3.2 that the performance metric is a function $\Lambda(\cdot)$ that maps a particular approximation model $\tilde{f}$ onto a positive scalar quantity, representing model quality. I.e., $\Lambda : S \mapsto \mathbb{R}^+$. In the SUMO Toolbox this is the task of the *Measure* class. Given a model object a *Measure* instance returns a scalar value indicating the quality of the model (lower is better). Again, different subclasses are possible, depending on the particular algorithm being implemented (cross validation, validation set, AIC, etc.). This is shown in figure 4.22. Each measure can also be configured with an error function ε. While Λ determines what algorithm is used to assess the model accuracy, ε determines what type of error is used (e.g., mean

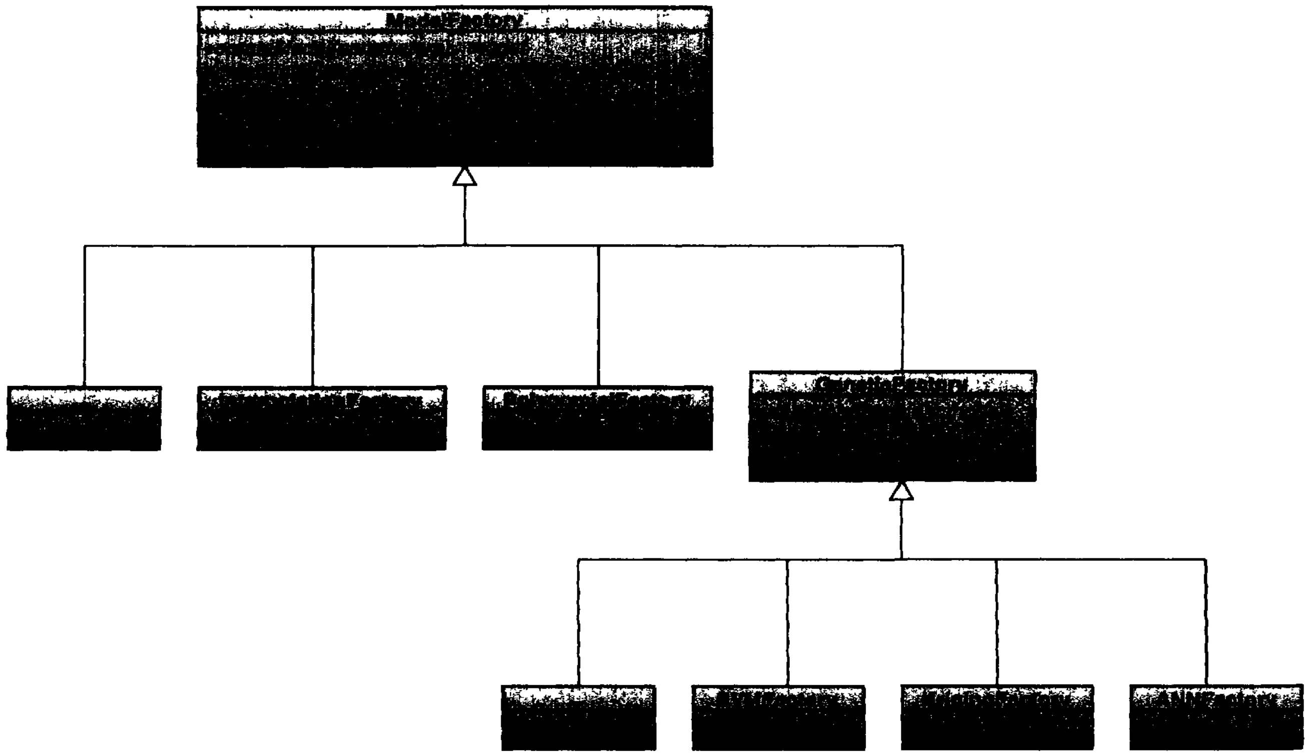


Figure 4.21: Model factory hierarchy

absolute error, maximum relative error, etc.).

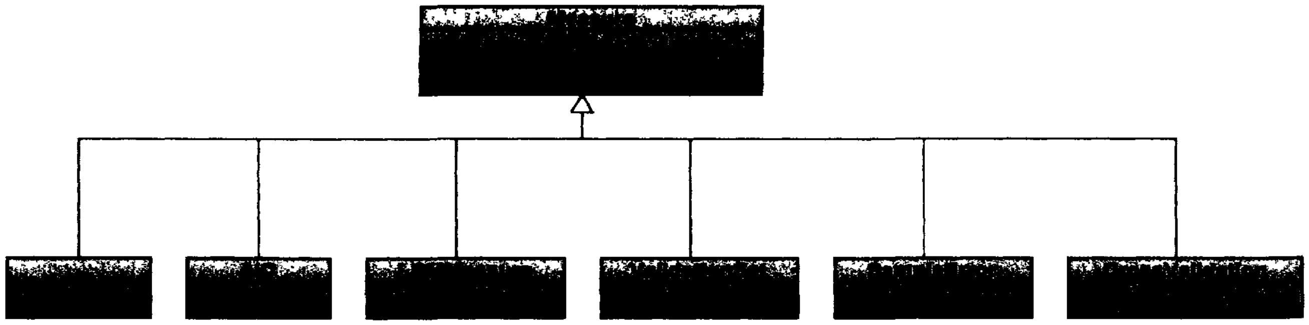


Figure 4.22: Measure hierarchy

The SUMO Toolbox also supports using multiple measure objects together. There are two options here:

- A single objective model builder is used: a weighted sum of the measures is minimized (each Measure object can be assigned a weight)
- A multi-objective model builder is used: all measures are minimized simultaneously (see chapter 8)

Finally, a measure can also be turned *off* or *on*. If set to *off* the measure is calculated on each model as usual, however its results are not used in the modeling process in any way. It is simply used as extra debug information. This is useful if you want to objectively track the true accuracy of the models produced using a dense test set without interfering with the model building process.

4.5.4.5 Model saving and elitism

While the model builder is going through the model complexity selection process it is important that the user is kept informed about the current progress and that intermediate results are saved for future reference. This is particularly relevant across multiple sampling iterations. For this purpose, and to ensure good models are not lost, the toolbox maintains a *best model trace*, i.e., a history of the k best models (as assigned by the model builder based on their measure score(s)) built so far. Each time a model is found that has a lower score than the previous best model it is saved to disk together with a plot of the model.

The purpose of the best model trace is as a kind of elite (to take a term from evolutionary computing). It is managed in one of two different ways depending whether

1. a single measure is enabled
2. multiple measures are enabled, either through scalarization or in a direct multi-objective approach

In the first case the best model trace is simply the linear ranking of the k best models, ordered according to their measure score. In the second case the best model trace is sorted according to the Pareto dominance of the different models on the different measures. In this case the best model trace contains the model with the lowest score on each measure separately (the extreme points of the Pareto front) plus the models taken from the first front, second front, third front, etc. until the maximum size k has been reached. By using such a scheme we prevent (to a degree) discarding models that perform very well on one measure but poorly on another. It might be that if a few more points become available, the model may turn out to score well on the other measure as well. The safety net is thus better than if a linear ranking were used.

4.5.4.6 Restart strategies

Recall from section 3.7.2 that the hyperparameter optimization surface is dynamic due to the non-stationary data distribution. This intuitively means that dynamic optimization algorithms like PSO should be preferred over static ones. This also implies that a restart strategy is needed to restart the model parameter optimization once the (static or dynamic) algorithm has converged. We defined five restart strategies: *continue* (simply continue with the final result of the previous modeling iteration), *random* (restart randomly), *model* (use model type specific information to decide where to start), and *intelligent* (a hybrid combination). Which restart strategy should be used is decided by the *AdaptiveModelBuilder* class.

The default restart strategy is *intelligent*, whose basic rationale is as follows: when only little data is available, there is more freedom for the model parameters to change but less so when the domain is more densely sampled. Thus, initially the search trace of a hyperparameter optimization algorithm (i.e., the points it has visited in the model parameter space) will not be very reliable[4]. As more data becomes available the reliability will increase. Or put differently, the confidence that a particular area of the hyperparameter optimization landscape leads to good models will increase.

Based on this rationale the intelligent restart strategy works as follows: as long as the optimization process is making improvements (the accuracy continues to increase) the hyperparameter optimization simply continues from the last solution when more data arrives. If this is not the case, a new starting point is selected based on the fuzzy combination of (1) information about what part of the model parameter space has been explored so far, (2) the regions of the parameter space that result in the best models, and (3), the current sample density (in order to calculate the reliability).

In this way, the next hyperparameter optimization iteration can be seeded optimally and the model parameter space is searched more thoroughly.

4.5.5 Configuration

All the subsystems and classes discussed so far are useless without some framework to instantiate and configure them. Thus, in order to allow full control over which subset of plugins should be used and to allow full customization of each plugin, the whole toolbox is extensively configurable using two XML files.

The first is the simulator configuration file and it defines the interface to the simulation code: number of input and output parameters, type of each parameter (real, discrete or complex), simulator executables and dependencies and/or one or more data sets. An example that describes the interface to simulation code for modeling a passive electrical component is shown in figure 4.23.

The second XML file contains the configuration of the toolbox itself: which outputs or inputs to model, model type to use, how the model quality should be assessed, whether sample evaluation should occur on a grid or cluster, etc. This file also allows the user to specify multiple *runs*. Recall from section 4.5.1 that every run can be configured separately and represents a different surrogate modeling experiment. Figure 4.24 shows part of an example toolbox configuration file that describes a possible setup for modeling the problem from figure 4.23. Figure 4.24 is an illustrative example that

[4]This of course depends on the metric that is used to guide the hyperparameter optimization. For example, say one uses a trustworthy metric that is independent of the available samples (e.g., a fixed dense validation set). In such an artificial setting (typically such information is not available) the reliability is maximal.

```
<Simulator>
   <Name>Step Discontinuity</Name>
   <Description>
      The stepdiscontinuity example contains a simulator
      and some datasets concerning a step discontinuity
      in a rectangular waveguide. Simulator code was written
      by Robert Lehmensiek and Petrie Meyer.
   </Description>

   <InputParameters>
      <Parameter name="frequency" type="real"/>
      <Parameter name="gapHeight" type="real"/>
      <Parameter name="stepLength" type="real"/>
   </InputParameters>

   <OutputParameters>
      <Parameter name="S11" type="complex"/>
      <Parameter name="S12" type="complex"/>
      <Parameter name="S21" type="complex"/>
      <Parameter name="S22" type="complex"/>
   </OutputParameters>

   <Implementation>
      <Executables>
         <Executable platform="matlab">
            StepDiscontinuity
         </Executable>
         <Executable platform="unix" arch="x86_64">
            /usr/local/bin/StepDiscontinuity
         </Executable>
      </Executables>

      <DataFiles>
         <ScatteredDataFile id="default">
            StepDiscontinuityScattered.txt
         </ScatteredDataFile>
      </DataFiles>
   </Implementation>
</Simulator>
```

Figure 4.23: Example simulator configuration file. This file defines the interface to the data source (number of inputs, outputs, implementation, etc.), in this case a modeling code from electronics.

shows how algorithms can be combined. It is not intended to be complete, in reality many more options and possibilities are available.

Figure 4.23 defined the output of the simulation code to be the four complex scattering parameters S_{11},S_{12},S_{21},S_{22}. In figure 4.24 one run is defined that models the first output twice using rational functions (once with added noise) and the second output using blind Kriging models. The suffix *_pso* implies the model complexity of the

```
<ToolboxConfiguration>
   <Plan>

     <!-- The identifiers between the tags are id's that refer to
          XML tags defined further down the file. These are not shown
          for brevity. Each of those tags are again fully configurable
          through their own set of options -->

     <InitialDesign>latinHypercube</InitialDesign>
     <SampleSelector>LOLA</SampleSelector>
     <Measure type="CrossValidation" target="0.01" errorFcn="mse"/>

     <!-- A run is a single surrogate modeling experiment.
     The configuration file may contain multiple runs. -->

     <Run name="StepDiscoDemo" repeat="1">

       <!-- Problem to model, refers to the file in figure 4.23 -->
       <Simulator>StepDiscontinuity</Simulator>

       <!-- How should each of the responses be modeled -->
       <Outputs>
          <Output name="S11">
             <AdaptiveModelBuilder>rational_pso</AdaptiveModelBuilder>
          </Output>

          <Output name="S11">
             <AdaptiveModelBuilder>rational_pso</AdaptiveModelBuilder>
             <Modifier type="noise" distribution="normal"/>
          </Output>

          <Output name="S12">
             <AdaptiveModelBuilder>blindKriging</AdaptiveModelBuilder>
             <SampleSelector>density</SampleSelector>
          </Output>

          <Output name="S11,S12" complexHandling="modulus">
             <AdaptiveModelBuilder>ann_genetic</AdaptiveModelBuilder>
             <Measure type="ValidationSet" target="0.01" />
          </Output>
       </Outputs>

     </Run>
    ...
   </ToolboxConfiguration>
```

Figure 4.24: Example toolbox configuration file. This file defines how the data source should be modeled: which outputs, which model type, how to score models, etc.

rational functions is optimized using PSO. The last output tag specifies that S_{11} and S_{12} should be modeled together in one ANN model (evolved using a genetic algorithm). As stated in the XML comments, the text between the opening and closing tags should be interpreted as an id that refers to a particular tag defined lower down in the file. This tag is again fully configurable through its own set of options. An example for the *ann_genetic* id is shown in figure 4.25.

Adaptive sampling is performed starting from an initial latin hypercube design, using the LOLA method [239] or density method (for blind Kriging). Also, since the ANN model type cannot handle complex numbers directly, the modulus is fitted instead. Since training an ANN is expensive, a validation error is used instead of the globally defined cross validation measure. Note that the user need not (unless he chooses otherwise) manually select a topology for the ANN, orders for the rational functions, or correlation parameters for the Blind Kriging model. These are determined automatically by the model builder.

```
<ToolboxConfiguration>
   <Plan>
  ...
   </Plan>
  ...
   <AdaptiveModelBuilder id="ann_genetic" type="GeneticModelBuilder"
                                               combineOutputs="true">
      <Option key="restartStrategy" value="continue"/>
      <Option key="crossoverFraction" value="0.7"/>
      <Option key="maxGenerations" value="10"/>
      <Option key="eliteCount" value="1"/>
      <ModelFactory id="ann" type="ANNFactory">
         <Option key="crossoverFcn" value="crossover"/>
         <Option key="mutationFcn" value="mutation"/>
         <Option key="creationFcn" value="createInitialPopulation"/>
         <!--initial hidden layer dimension-->
         <Option key="initialSize" value="3,3"/>
         <!--comma separated list of allowed learning rules-->
         <Option key="allowedLearningRules" value="trainbr"/>
         <!--how many epochs to train for-->
         <Option key="epochs" value="300"/>
      </ModelFactory>
   </AdaptiveModelBuilder>
   ...
</ToolboxConfiguration>
```

Figure 4.25: Example of the configurable component ann_genetic that is referred to in figure 4.24. The component represents an adaptive model builder that implements a genetic algorithm and drives a neural network model factory.

4.5.6 Optimizer hierarchy

Optimization plays an important role in the toolbox with many modeling and sampling algorithms reducing to an optimization problem. Therefore it is useful if a pluggable optimization framework is available that allows one to quickly try and compare different optimization algorithms. For this reason the toolbox includes a *Optimizer* class that forms the basis of a lightweight, general purpose optimization hierarchy (figure 4.26).

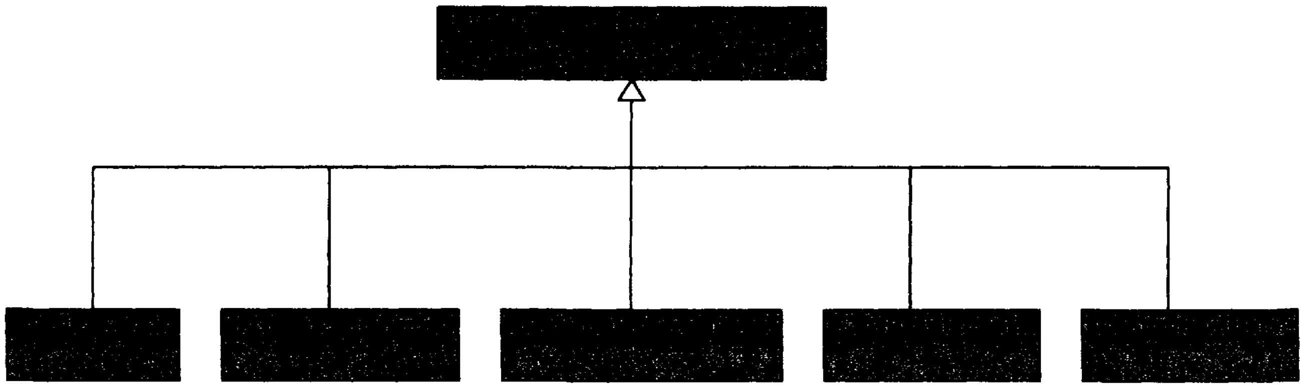


Figure 4.26: Optimizer hierarcy

Different subclasses are available that implement various optimization algorithms and that can be reused inside model builders, sample selectors, measures, or any other component.

4.5.7 Logging and Profiling

During the execution of the toolbox it is important that all decisions are properly logged and that the modeling progress can be monitored in real time. This is important to facilitate debugging but also helps a domain expert gain trust in the algorithms. The sumo toolbox distinguishes between two types of logging information: textual and numeric.

Textual logging information is handled through the standard *java.util.logging* framework (figure 4.27). Central is the *Logger* object on which logging calls are made. Loggers are organized in a hierarchical namespace and child Loggers may inherit some logging properties from their parents in the namespace. These logger objects allocate *LogRecord* objects which are passed to *Handler* objects for publication. Both loggers and handlers may use logging levels (FINE, INFO, WARNING, SEVERE, ...) and (optionally) *Filter* objects to decide if they are interested in a particular record. When it is necessary to publish a *LogRecord* externally, a handler can (optionally) use a *Formatter* to localize and format the message before publishing it to an I/O stream. Each logger keeps track of a set of output handlers, which direct output to a file on disk, the console, a remote machine, to another handler, etc.

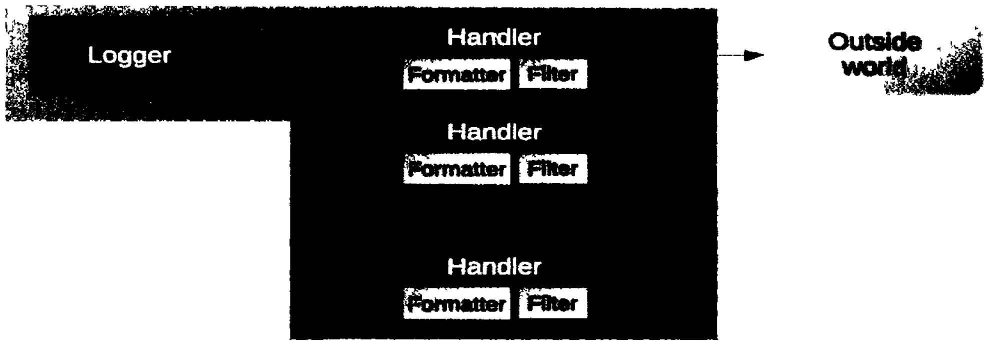


Figure 4.27: Logging subsystem based on java.util.logging

The *java.util.logging* APIs are structured so that calls on the logger APIs can be cheap when logging is disabled. If logging is disabled for a given log level, then the logger can make a cheap comparison test and return. If logging is enabled for a given log level, the logger is still careful to minimize costs before passing the record into the handlers. In particular, localization and formatting (which are relatively expensive) are deferred until the handler requests them.

For numeric data a custom Profiler framework is available (figure 4.28). Central is the *Profiler* class which essentially implements a data table where each column represents a variable that is monitored. The *Profiler* class contains methods for publishing *ProfileRecord* objects, each of which represents a data tuple, or row, of the data table. Profilers are organized in a linear namespace and managed through a *ProfilerManager* singleton object. Analogous to the logging framework each profiler can have multiple *OutputHandler*'s that can direct data to an ASCII file, Java *JTable*, or a chart generation component. The latter renders the data as a chart in order to save it as an image and/or display it in real time in the profiler browser GUI. Other handlers can of course be freely defined, e.g., a handler that saves output directly in a database or a handler that is able to pipe data directly over the network to a remote client. Like the logging framework, a profiler can be enabled or disabled if necessary.

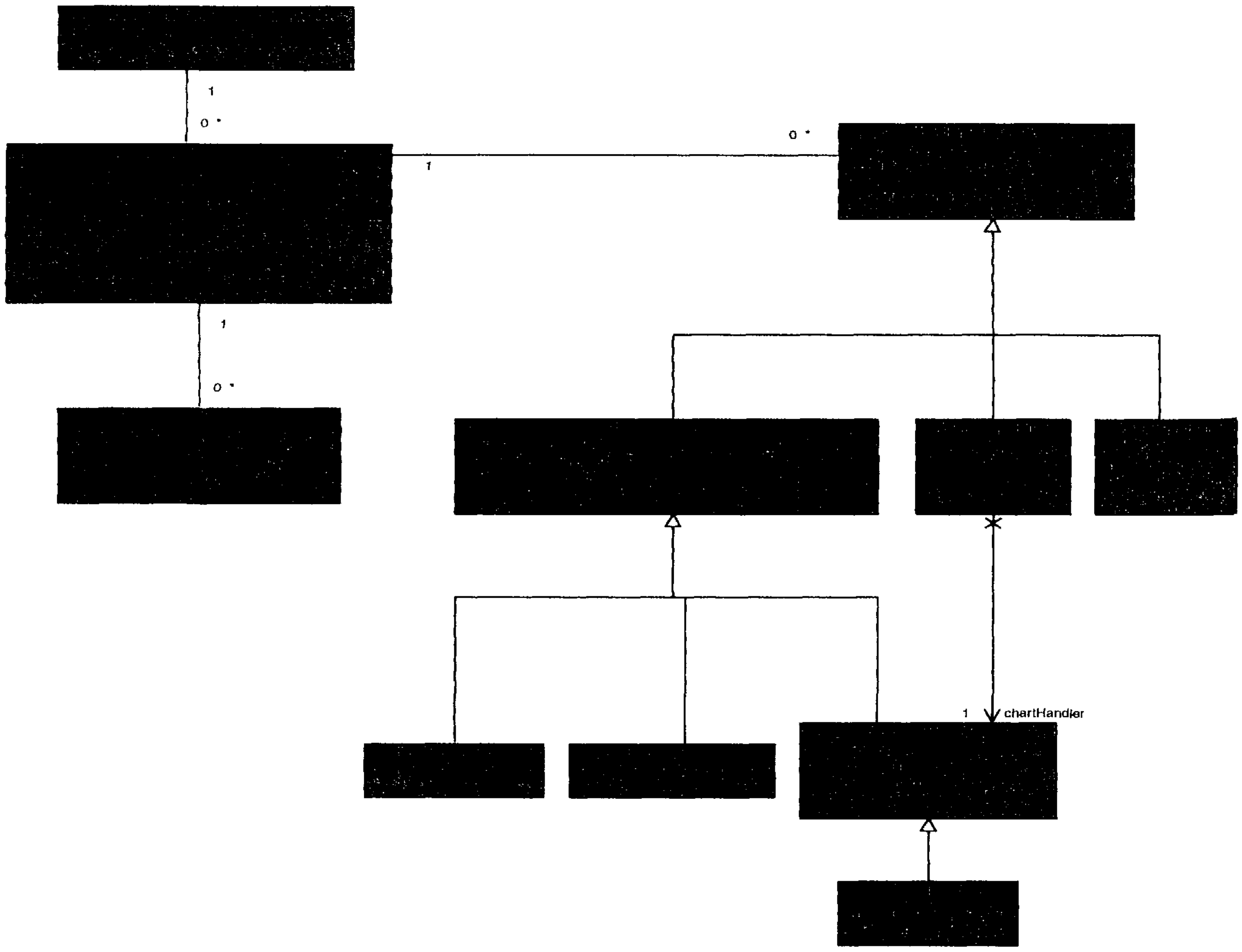


Figure 4.28: Profiling subsystem

A screenshot of the profiler browser GUI in its current incarnation is shown in figure 4.29. Examples of profilers include:

- a profiler to keep track of the model parameters during the hyperparameter opti-

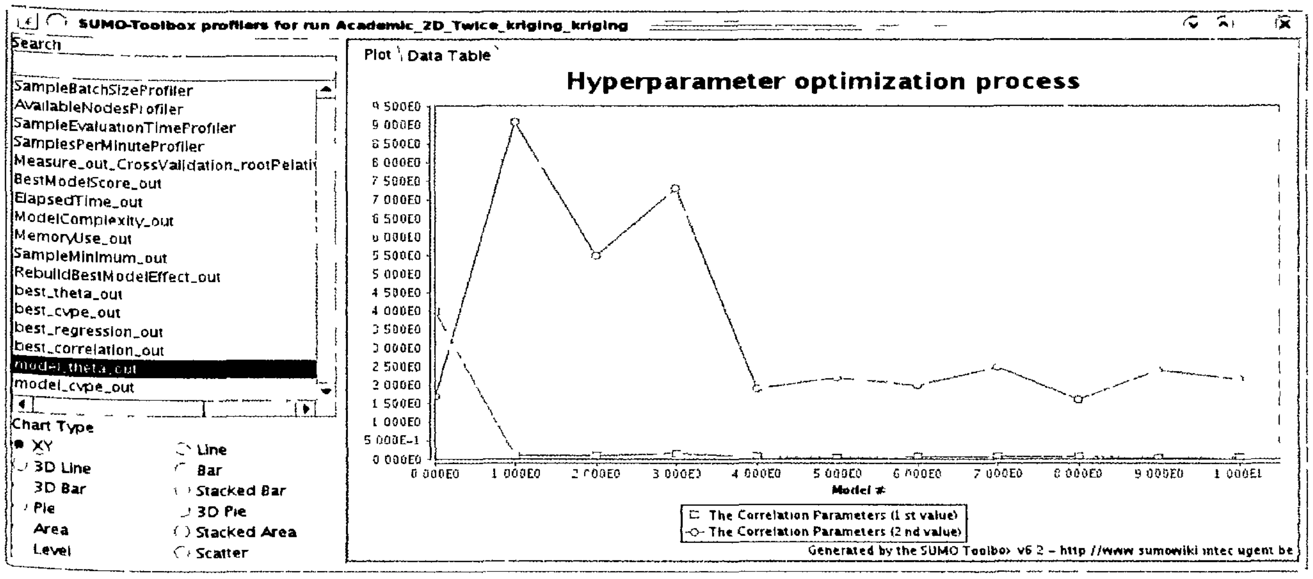


Figure 4.29: Screenshot of the current profiler browser GUI. Each enabled profiler registers itself with this browser which shows and updates the data in real time.

mization (e.g., how does the regularization parameter γ evolve versus the number of samples as we optimize the SVM hyperparameters using pattern search).

- a profiler that keeps track of the average duration of a single simulation
- a profiler that keeps track of the accuracy of the current best model
- a profiler that plots the Pareto search trace during a 2-objective hyperparameter optimization

4.6 What about optimization?

Recall from section 1.1.1 that the focus of this dissertation is on accurate models of the complete design space, i.e., global surrogate models. Most research on surrogate models is, however, directed towards optimizing the response instead of mapping it globally. While the SUMO Toolbox does not implement pure SBO methods it does provide a flexible implementation of infill based algorithms. The most popular of which is probably the Efficient Global Optimization (EGO) algorithm [89]. Infill algorithms like EGO, are neither purely local or global. Rather, they use a global model of the design space together with a sampling strategy that focuses on sampling near the predicted optima (depending on the criteria used). The infill framework and its relation to the toolbox is discussed in the following two subsections.

4.6.1 Optimization through infill criteria

Optimization methods can greatly benefit by taking advantage of surrogate models. The extra information gained helps in avoiding local optima and efficiently guiding the search to global optima. E.g., a surrogate model offers a cheap alternative to the expensive simulator for exploration and exploitation purposes. Likewise, the practitioner is able to explore the region of the final optimum quite easily. Various directions have been taken to incorporate surrogate models in the optimization process, good reviews can be found in [4,44,63].

A recent approach is to use global surrogate models and emphasize more on adaptive sampling versus local model management strategies. However note that while these surrogate models are a global approximation, they are not necessarily accurate over the whole design space. This depends on the adaptive sampling strategy, or infill criteria. Starting from an initial approximation of the design space, the infill criterion identifies new samples of interest (infill or update points) to update the approximation model. It is crucial in global SBO to strike a correct balance between exploration[5] and exploitation[6]. A well-known infill criterion that is able to effectively solve this trade-off is expected improvement. Going back to work by Mockus et. al. in 1978 1978 [248] it is most well known through the EGO algorithm proposed by Jones et al. [89,249]. Subsequently, Sasena compared different infill criteria for optimization and investigated extensions of those infill criteria for constrained optimization problems in [250].

Thus, in sum, an unkown function can be optimized by repeatedly sampling a surrogate model of the function in the neighborhood of the minimum. The criterion used to drive the sampling is called an infill criterion, of which the expected improvement cirterion is the most well known example. The expected improvement equation (4.4) is easiest to interpret graphically, as in figure 4.30. At $\mathbf{x} = 0.5$, a Gaussian probability density function is drawn and expresses the uncertainty about the predicted function value of a sampled and unknown function $y = f(\mathbf{x})$. Thus, the uncertainty at any point $\mathbf{x}$ is treated as the realization of a random variable $Y(\mathbf{x})$ with mean $\hat{y} = \hat{f}(\mathbf{x})$ (prediction) and variance $\hat{s}^2 = \hat{\sigma}^2(\mathbf{x})$ (prediction variance). Assuming the random variable $Y(\mathbf{x})$ is normally distributed, then the shaded area under the Gaussian probability density function is the Probability of Improvement (PoI) of any newly calculated function value $y(\mathbf{x})$ over the intermediate minimum function value f_{min}(the dotted line), denoted as $P(y \leq f_{min})$, i.e.,

[5]enhancing the general accuracy of the surrogate model

[6]enhancing the accuracy of the surrogate model solely in the region of the (current) optimum

$$PoI = P(y \leq f_{min}) = \int_{-\infty}^{f_{min}} Y(\mathbf{x})\,dy = \Phi\left(\frac{f_{min} - \hat{y}}{\hat{s}}\right), \tag{4.1}$$

where $\Phi(t)$ is the standard normal cumulative distribution function $\Phi(t) = \frac{1}{2}\left[1 + erf\left(\frac{t}{\sqrt{2}}\right)\right]$ and $erf(\cdot)$ is the error function. The probability of improvement is already a very useful infill criterion. However, while this criterion describes the possibility of a better minimum function value, it does not quantify how large this improvement will be.

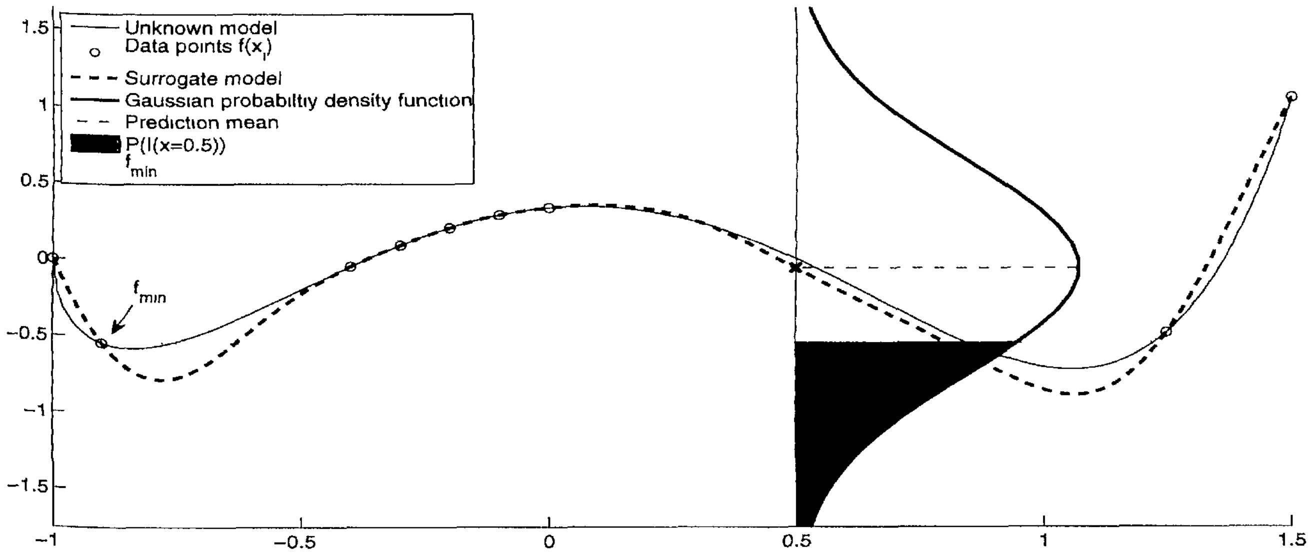


Figure 4.30: Graphical illustration of a Gaussian Process and expected improvement. A surrogate model (dashed line) is constructed based on some data points (circles). For each point the surrogate model predicts a Gaussian probability density function (PDF). At $x = 0.5$ an example of such a PDF is drawn. The volume of the shaded area is the probability of improvement and the first moment of this area is the expected improvement.

EI quantifies the improvement by considering the first moment of the shaded area, i.e., every possible improvement over f_{min} multiplied by the associated likelihood. For continuous functions EI is an integral defined as:

$$E(I) = \int_{-\infty}^{f_{min}} I \cdot Y(\mathbf{x})\,dy, \tag{4.2}$$

where

$$I = max(f_{min} - y, 0). \tag{4.3}$$

Hence, EI can be rewritten in closed form as:

$$E(I) = \begin{cases} (f_{min} - \hat{y}) \cdot \Phi\left(\frac{f_{min}-\hat{y}}{\hat{s}}\right) + \hat{s} \cdot \phi\left(\frac{f_{min}-\hat{y}}{\hat{s}}\right) & if\ \hat{s} > 0 \\ 0 & if\ \hat{s} = 0 \end{cases}, \qquad (4.4)$$

where $\phi(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}$ is the standard normal probability density function and $\Phi(\cdot)$ is the cumulative distribution function defined before. Here we ignore the bias in the prediction variance of kriging, details on how to estimate the correct kriging variance by bootstrapping is given in [251]. The EI (Equation 4.4) and PoI (Equation 4.1) serve as utility functions, often conceived as figures of merit, which have to be optimized over $\mathbf{x}$ to find the subsequent data point to evaluate. Note, however, that besides the prediction $\hat{y} = \hat{f}(\mathbf{x})$ of the surrogate model, a point-wise error estimation $\hat{s} = \hat{\sigma}(\mathbf{x})$ of the surrogate is also required.

Therefore, the original EGO algorithm used kriging as surrogate model of choice, as Kriging provides closed formulae for prediction as well as a point-wise error estimation [6, 252]. Kriging, in fact, is part of a broader class of approximation methods, namely, Gaussian Processes (GP). While traditional approximation methods only predict a single function value, GP methods predict a complete normal distribution $Y(\mathbf{x}) \sim \mathcal{N}(\hat{y}, \hat{s})$ for each point $\mathbf{x}$. The predicted distribution imparts the probability that a particular function value occurs. For a full overview of modern GP the reader is referred to the excellent GP reference book of Rasmussen et al. [127]. As a final note, in case there is a cheaper, but more inaccurate, simulator available multi-fidelity approximation models can be used. For instance, co-Kriging is able to incorporate several levels of fidelity samples in its prediction [253, 254]. This has been successfully used to optimize a transonic civil aircraft wing in [255].

Over the years small changes and improvements to expected improvement have been suggested, see for example Sobester et al. [256]. Parallel versions of EGO have also been proposed [257, 258], as have multi-objective formulations [259–262]. While expected improvement is a very popular criterion, other infill criteria were suggested that solve the exploitation and exploration trade-off completely differently. For instance, Regis et al. introduced the Constrained Optimization using Response Surfaces method in [263]. In addition, instead of optimizing the infill criteria other, less expensive, search strategies have been developed. Wang et al., for instance, treat the infill criterion as a probability density function and propose a dimensional-free method to efficiently sample according to the probability given by the infill criterion. This has been implemented by considering the prediction itself as the infill criterion in [240, 264], thus focusing only on exploitation, though other infill criteria may be used. More information on infill criteria can be found in [8] and the references therein.

4.6.2 Infill criteria in the SUMO Toolbox

As should be clear, the infill criteria match up perfectly with the *OptimizeCriterion* sample selector described in section 4.5.2. So applying infill based optimization methods simply requires the use of a specific sample selector. A number of criteria are available that can be used to drive infill point selection process. These include (weighted) expected improvement, probability of improvement and generalized expected improvement.

4.7 Applying the toolbox

Given the many subsystems and frameworks in the toolbox it is useful to take a step back and briefly explain the steps a user must go through in order to generate a model for a particular problem. We only list the main steps here, details can be found on the SUMO Toolbox website.

1. *Comply with the expected data format*: firstly the user should assure his data source (simulator, data file) adheres to the simple format expected by SUMO (e.g., a column based ASCII format with one point per row).
2. *Generate a simulator configfile*: secondly the user needs to generate a simulator configuration file (cfr. the example in figure 4.23) and place all files relevant to the data source (documentation, shell scripts, etc.) into a project directory with the same name as the simulator file (without the extension).
3. *Select appropriate options*: starting from the default toolbox configuration the user selects the appropriate plugins and options for his problem.
4. *Perform the run*: execute the toolbox with the modified toolbox configuration file, monitor the modeling process and analyze the results

Thus applying the toolbox to a novel problem is rather straightforward. Difficulties usually lie in ensuring the simulation code is well behaved (e.g., terminates properly, returns valid results, license environment is properly setup, etc.). Steps (2) and (3) currently require manually editing the XML files. Ideally this should be avoided but the structure of both files is still simple enough to grasp and modify manually. A prototype GUI for both files is available but not yet robust enough for general use.

4.8 Extending the toolbox

Besides applying the toolbox in its existing form to new problems, an important use case is extending the toolbox to include new algorithms and techniques. The only important prerequisite here is that the user understands the basic concepts behind object oriented programming. If this is the case, extending the toolbox can be done quite easily by subclassing a particular base class.

For example, to add a new DoE the user should write a Matlab file that subclasses from *InitialDesign*, implement his logic in the *generate(..)* method and add a configuration section with some options to the toolbox configuration file. The same is true for adding a new measure, sample selector, optimizer, or logging/profiler handler.

Adding support for a new approximation method is a bit more work since it involves subclassing two classes (*Model* and *ModelFactory*) instead of one. The minimum effort amounts to:

- Subclassing *Model* and providing two Matlab functions:
  - *constructInModelSpace(..)*: fits the new fitting technique on the given points. This method may be omitted if the fitting technique relies on lazy learning.
  - *evaluateInModelSpace(..)*: evaluates the new fitting technique on the given points.
- Subclassing *ModelFactory* and implementing the *createModel(..)* method which simply returns an instance of the *Model* subclass encapsulating the new fitting technique.

With these two files the SUMO Toolbox will be able to use the new fitting technique when modeling data. Hyperparamter optimization is, however, not yet possible. For that to be possible the user should also implement the following two methods of *ModelFactory*:

- *createInitialModels(..)*: return an array of model objects with the parameters set to the passed values
- *getBounds(..)*: return the bounds of each model parameter

With these additional two functions all of the hyperparameter optimization algorithms (GA, PSO, simulated annealing, EGO, NSGA-II, etc.) available in the toolbox can be used to select the model parameters of the new fitting technique.

In sum, if the user is aware of the basics of object oriented design it becomes straightforward to extend the toolbox with new techniques. The new plugin will also automatically be able to make use of the other frameworks available (visualization, optimization, data interfacing, etc.). The overall advantage is that a user can concentrate on the algorithm itself, and need not worry about the associated plumbing.

4.9 What about classification or time series prediction?

Section 1.1.2 outlined the scope of this dissertation and there it was explicitly stated that we are only concerned with function approximation type problems. Not classification or time series prediction. However, throughout the discussion so far, readers familiar with either problem will have recognized that many of the problems and approaches discussed in this dissertation apply just as well to classification and time series prediction as they do to regression. Examples are the problems of model selection and hyperparameter optimization.

It turns out that while the SUMO Toolbox finds its main use cases in regression, the abstractions that were developed as part of the framework can just as easily be extended to support classification or time series prediction. To give a concrete example, adding the necessary functionality so that the full SUMO framework could generate a classification model required only 30 minutes of work[7]. While of course some extra polishing is needed it is a testament to the clean and generic design of the framework. Similar support could be implemented for time series prediction if this is needed.

A classification example is given in section 10.10, but we will not dwell on these topics any further. A demo configuration file is available as part of the SUMO distribution that illustrates the classification support and that will used as a basis for future work in this area.

4.10 Critique

Presenting and advocating a new approach/framework to solve a particular problem is one thing. Equally important is to regard ones approach critically in order to identify flaws or limitations (keeping in mind the scope).

Let us start with an important advantage of the approach taken by the SUMO Toolbox: the toolbox provides a very convenient and feature rich platform that can be leveraged to quickly implement new algorithms and try out ideas. The infrastructure plumbing (model plotting, saving, logging, profilers, optimization, etc.) has been taken care of, a user of the platform can focus purely on the algorithm or technique of interest. At the same time a whole suite of plugins is available to easily test and compare with. The downside is that this results in increased complexity of the underlying framework and that great care must be taken when designing the necessary APIs. Sacrificing software engineering design principles during this process is a recipe for disaster in the long term. Thus a considerable amount of thought, discipline, and stewardship is needed to ensure the stability, robustness, and flexibility of the underlying code base. In addition, maintaining a reusable component library and ensuring the software developers can use this library can be expensive [237]. Care must be taken that all plugins remain up to date. These are of course problems that every non-trivial software engineering effort must face. Thus the 'solutions' are well known and described: regular communication between developers, code and design documentation and automated (regression) testing.

[7]`http://sumolab.blogspot.com/2009/10/surrogate-models-for-classification.html`

Secondly, while the SUMO Toolbox makes it very easy to try and compare different surrogate modeling methods it does not really solve the fundamental problem of which configuration of plugins should be used for a particular problem. In the ideal case a second level of adaptivity is available that makes the selection of components itself an automatic step. Note, though, that such additional level is only possible if an underlying framework with the necessary abstractions is already available. This is the domain of meta-learning [265]. The integrated hyperparameter optimization is already a good step in that direction. As is the evolutionary model type selection (chapter 7), multi-objective modeling framework (chapter 8), and the ongoing work on dynamic sample selection strategies. However, an important downside of additional adaptivity and automation is that it comes at a cost. Each increment of adaptiveness brings with it new parameters that must be set (increasing the computational cost), adds another layer of complexity and indirection, and relinquishes user control. Einstein famously said "*Make everything as simple as possible, but not simpler*". Therefore a very important design goal of the toolbox presented in this chapter was to ensure that adaptive algorithms played an important role, but full manual control and problem specificity was not sacrificed. As Keane and Nair put it in [4]: "*it must also be remembered that even the most complex approaches are still subject to the "no free lunch" theorems, in that, by adopting highly sophisticated approaches, one is commonly sacrificing generality of application. [...] sometimes relatively inefficient methods may be preferred because they are simpler to setup or manage*". Thus, while an extra layer of adaptivity may be intuitively attractive and useful its net benefit should not be overestimated.

A limitation of the toolbox in its current incarnation is that it does not provide a formalized framework for the integration of coarse domain knowledge. Often rudimentary equations or approximations describing the problem are available and should be used in the modeling process. While such knowledge can be integrated into the SUMO Toolbox through new plugins there are no formalized interface primitives or step by step instructions to facilitate the process. Ideally the necessary APIs should be added to facilitate knowledge integration through the standard space mapping and knowledge integration methods [74–76]. There are no inherent limitations in the current framework that prevent this from being implemented. Rather it is a question of time and manpower. This is a topic for future work. The same is true for a user friendly API and plugin system for parameter screening methods. Barthelemy and Haftka [266] state that the increase in cost for generating a response surface as a function of the number of input variables is quadratic. The author is inclined to say it is even more than that. Thus any integration with standard screening methods would be very useful.

Another critique is that Moore's Law and the availability of HPC resources weakens the argument in favor of sequential design (and metamodeling in general). If enough resources are available a brute force approach, though less elegant, can easily be used. However, while evolutions in computing power have greatly eased the modeling and simulation problem, the complexity of the underlying algorithms and the drive to finer timescales and increased dimensionality, readily balance gains in computational power. Renowned nuclear physicist Edward Teller once stated "*A state-of-the-art calculation requires 100 hours of CPU time on the state-of-the-art computer, independent of the decade.*". Furthermore, the use of HPC resources still incurs a cost in resources. Thus every simulation saved through an adaptive approach is a simulation earned.

Finally, there is no guarantee that an adaptive, self tuning approach will always outperform a simple full factorial design together with a slightly hand tuned model. In some cases the benefit of using the former may simply be nihil or too small to be worth it. Likewise, high accuracy is not always needed. Often engineers are perfectly happy with the one or two significant digits that a straightforward polynomial regression can give them. These criticisms are of course valid, ultimately the decision lies with the domain expert and the resources at his/her disposal. The mantra should always be "*use the right tool for the right job*" and if in doubt apply Occams razor.

4.11 Conclusion

This chapter discussed the philosophy and architecture of a concrete implementation of the ideas from chapter 3. The SUMO Toolbox is presented as an adaptive tool that integrates different modeling approaches and implements an automated, global surrogate model construction algorithm. Given a simulation engine or other data source, the toolbox generates a surrogate model within the predefined accuracy and resource limits set by the user. In order to cope with the heterogeneity of problems and existing methods, strong focus is placed on a modular extensible design without being too cumbersome to use or configure. The concrete design goals were: flexibility (many problems/domains), portability (easily integrate with the design chain), extensibility (many techniques), adaptivity (automation), allow for full manual control (configuration), and extensive logging (traceability).

Given this design philosophy, the toolbox caters to both the scientists working on novel surrogate modeling techniques as well as to the engineers and domain experts who need the surrogate model as part of their overall design process. For the former, the toolbox provides a common platform on which to deploy, test, and compare new modeling algorithms and sampling techniques. For the latter, the software functions as a highly configurable and flexible component to which surrogate model construction can be delegated. In this sense the toolbox can also be used as a model generator backend in projects like DAKOTA, VisualDOC, Geodise, and others. The hope is that the availability of such a tool will lower the barrier of entry for domain experts to advanced modeling and sampling techniques and facilitate the transfer of knowledge from surrogate modeling researchers.

5 Case Study: Low Noise Amplifier

The path of precept is long, that of example short and effectual.

– Seneca, Roman philosopher

5.1 Introduction

We now present a concrete case study that is a good illustration of why and how the SUMO Toolbox may be used. It concerns an investigation into the feasibility of generating replacement metamodels for a Low Noise Amplifier (LNA). Previous investigations [142, 267] have shown ANNs to be very promising for modeling admittance functions and noise figures when two or three design variables are considered, achieving high accuracy for a minimal number of data points. Thus, the motivation for this case study is to investigate if these good results are maintained as the problem difficulty and number of design variables is increased. In particular we wish to study how the accuracy of neural models scale in function of the number of design variables and number of data points. In addition we wish to investigate how this relationship compares with other popular surrogate model types such as rational functions, Support Vector Machines (SVM), Radial Basis Function (RBF) models, and Kriging models.

5.2 Background

An LNA is an electronic amplifier used to amplify very weak signals (for example, captured by an antenna). It is usually located very close to the detection device and is frequently used in microwave systems like GPS. Thus an LNA is the typical first stage of a receiver, having the main function of providing the gain needed to suppress the noise of subsequent stages, such as a mixer. In addition it has to give negligible distortion to the signal while adding as little noise as possible [268]. The performance figures of an LNA (gain, input impedance, noise figure and power consumption) can be determined by means of computer simulations where the underlying physics is accurately taken into account. Each simulation typically requires about a minute, too long for a circuit designer to work with efficiently. Instead a designer could use a very accurate surrogate model (based on circuit simulations) to quickly explore how performance figures of the LNA scale with key circuit-design parameters, such as the dimensions of transistors, passive components, signal properties and bias conditions. The goal of the design process is to figure out one or more sets of design parameters resulting in a circuit which fulfills the specifications, i.e., constraints given on the performances.

Obtaining the required circuit design parameters can be done through an approximation of the circuit performance figures based on one or more surrogate models. This is referred to as 'forward model' of the circuit. A forward model can be either obtained via direct modeling of circuit performances (i.e., a one step approach) or by using intermediate surrogate models of a convenient set of behavioural parameters (e.g. admittances and noise functions) and compute performances via analytical equations in a post-processing step (i.e., a two step approach). This is illustrated in figure 5.1.

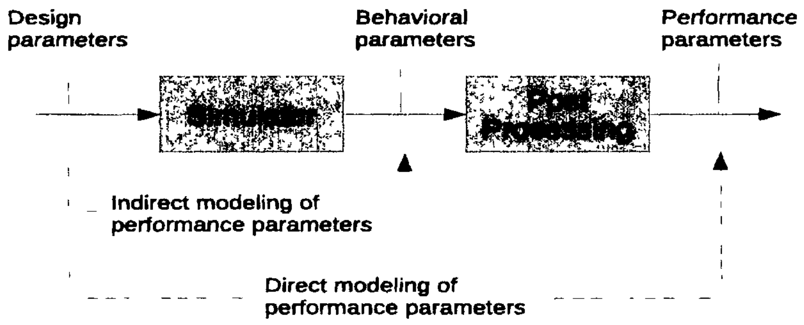

Design parameters
Behavioral parameters
Performance parameters
Simulator
Post Processing
Indirect modeling of performance parameters
Direct modeling of performance parameters

Figure 5.1: Direct and indirect modeling of the LNA performance parameters

In this chapter we are only concerned with the modeling of the behavioral parameters (indirect approach). The direct approach will be revisited in chapter 9.

For this study, in order to keep the computation times manageable, we replace the expensive simulations by a first-order analytic model. This allows us to do a full modeling study. The results and conclusions of the case study are not affected by the use of analytical functions since the qualitative behavior is the same. Once the necessary insights have been gained, the next step is to model the simulation code directly.

The functions that govern the approximate behavior of the LNA are given by equations 5.1 to 5.9. The small signal representation that was used to derive these equations is shown in figure 5.2.

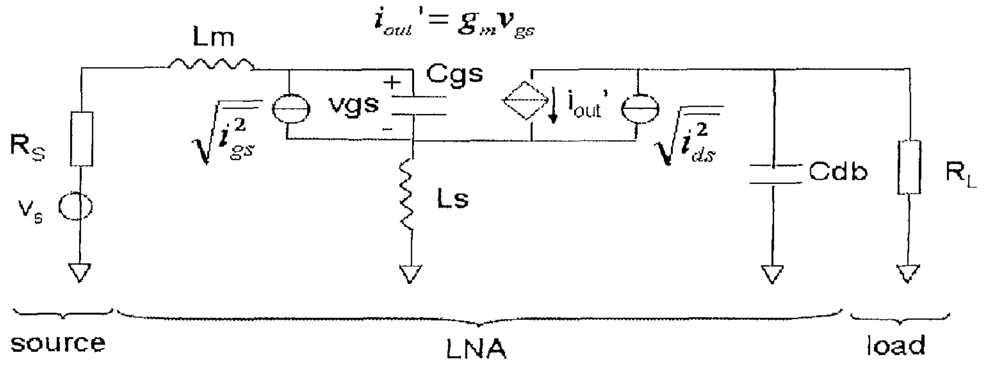


Figure 5.2: Small signal representation of the LNA

The input parameters are (in order of significance): the MOSFET width W, the inductance L_s, the frequency f, the MOSFET length L, the voltage V_{GT}, and the inductance L_m, with $W = 100 \cdot 10^{-6} \cdot 10^{W_n}$ m, $L_s = 0.5 \cdot 10^{-9} \cdot 10^{L_{sn}}$ H, $f = (11 + 10 \cdot f_n) \cdot 10^9$ Hz, $L = (90 + 30 \cdot L_n) \cdot 10^{-9}$ m, $V_{GT} = 0.275 + 0.2 \cdot V_{GTn}$ V, and $L_m = 1 \cdot 10^{-9} \cdot 10^{L_{mn}}$ H. Where W_n, L_{sn}, f_n, L_n, V_{GTn}, L_{mn} are normalized to [-1 1]. If a parameter is not used it is clamped to the middle of its domain. The output parameters are the admittances y_{11},y_{12}, the noise parameters $\sqrt{\overline{i_{in}^2}}$, $\sqrt{\overline{i_{out}^2}}$ and their correlation ρ. The higher order y-parameters (y_{111},y_{121},...) were not considered for this study. The approximate admittances are given by:

$$y_{11} \cong \frac{j\omega C_{gs}}{1 - \omega^2 C_{gs}(L_s + L_m) + j\omega L_s g_m} \tag{5.1}$$

$$y_{12} \cong \frac{g_m}{1 - \omega^2 C_{gs}(L_s + L_m) + j\omega L_s g_m} \tag{5.2}$$

The approximate input noise-current ($\sqrt{\overline{i_{in}^2}}$), output noise-current ($\sqrt{\overline{i_{out}^2}}$), and their correlation (ρ) are defined as:

$$f_{gs,in} = \frac{1 + j\omega L_s g_m}{1 - \omega^2 C_{gs}(L_s + L_m) + j\omega L_s g_m} \tag{5.3}$$

$$f_{ds,in} = \frac{\omega^2 C_{gs} L_s}{1 - \omega^2 C_{gs}(L_s + L_m) + j\omega L_s g_m} \tag{5.4}$$

$$f_{ds,out} = \frac{1 - \omega^2 C_{gs}(L_s + L_m)}{1 - \omega^2 C_{gs}(L_s + L_m) + j\omega L_s g_m} \tag{5.5}$$

$$f_{gs,in} = \frac{1 + j\omega L_s g_m}{1 - \omega^2 C_{gs}(L_s + L_m) + j\omega L_s g_m} \tag{5.6}$$

$$\sqrt{\overline{i_{in}^2}} = \sqrt{|f_{gs,in}|^2 \cdot \overline{i_{gs}^2} + |f_{ds,in}|^2 \cdot \overline{i_{ds}^2} - 2 \cdot \mathrm{Im}(0.4 f_{gs,in} f_{ds,in}^*) \sqrt{\overline{i_{gs}^2} \cdot \overline{i_{ds}^2}}} \tag{5.7}$$

$$\sqrt{\overline{i_{out}^2}} = \sqrt{|f_{gs,out}|^2 \cdot \overline{i_{gs}^2} + |f_{ds,out}|^2 \cdot \overline{i_{ds}^2} - 2 \cdot \mathrm{Im}(0.4 f_{gs,out} f_{ds,out}^*) \sqrt{\overline{i_{gs}^2} \cdot \overline{i_{ds}^2}}} \tag{5.8}$$

$$\rho = \frac{f_{gs,in} f_{gs,out}^* \cdot \overline{i_{gs}^2} + f_{ds,in} f_{ds,out}^* \cdot \overline{i_{ds}^2} + 0.4j \cdot (f_{gs,in} f_{ds,out}^* - f_{gs,out}^* f_{ds,in}) \sqrt{\overline{i_{gs}^2} \cdot \overline{i_{ds}^2}}}{\sqrt{\overline{i_{in}^2} \cdot \overline{i_{out}^2}}} \tag{5.9}$$

The remaining parameters have been set to fixed, typical values: $\omega = 2\pi f$, $g_m = 1 \cdot 10^{-4} \frac{W}{L} V_{GT}\,\mathrm{AV}^{-1}$, $C_{GS} = 0.01 \cdot WL\,\mathrm{F}$, $\overline{i_{gs}^2} = 2 \cdot 10^{-3} \frac{W}{L}\,\mathrm{pA^2Hz^{-1}}$, $\overline{i_{ds}^2} = 0.5 \frac{W}{L}\,\mathrm{pA^2Hz^{-1}}$. The meanings are as follows: ω is the angular signal frequency , g_m is the MOSFET transconductance, C_{GS} is the gate-source capacitance, $\overline{i_{gs}^2}$ is the gate-source noise current spectral density of the MOSFET, and $\overline{i_{ds}^2}$ is the drain source noise current spectral density of the MOSFET.

A study of $\sqrt{\overline{i_{in}^2}}$ in the 2D case (focusing largely on Kriging) was carried out in [142], a more extensive study (covering $\sqrt{\overline{i_{in}^2}}, \sqrt{\overline{i_{out}^2}}, y_{11}, y_{12}$ and multiple model types) for the 3D case was done in [267]. Building on those previous results we now include all output parameters, increase the maximal number of input parameters to 6, and increase the maximum number of data points to 1500.

5.3 Experimental setup

For this problem, the model types we consider are ANNs, rational functions, RBF models, Least Squares SVMs (LS-SVM, implementation from [269]) and Kriging models (implementation from [270]). The ANN models are based on the Matlab Neural Network Toolbox and are trained with Levenberg Marquard backpropagation with Bayesian regularization [271,272] (300 epochs). The topology and initial weights are determined by a Genetic Algorithm (GA). The complexity of the rational functions is determined by two algorithms, a custom stochastic hill climber (HC) or a GA. A GA is also used to set the regression/correlation functions of the RBF models, the correlation parameters of the Kriging models (the regression function is set to linear and the correlation function to Gaussian) and the hyperparameters of the LS-SVM models (an RBF kernel is used). Both the LS-SVM and Kriging hyperparameters are searched in log_{10} space with $\sigma \in [-4,4]$, $\gamma \in [-5,5]$ and $\theta_i \in [-5,3]$.

For the complex outputs (y_{11}, y_{12}, ρ) rational and RBF models are used to model the complex data directly while ANN, LS-SVM and Kriging are used to model the real and imaginary parts separately in separate models (since the implementations available only work in $\mathbb{R}$). For the noise outputs ($\sqrt{i_{in}^2}, \sqrt{i_{out}^2}$) rational functions, ANN, LS-SVM and Kriging are used. The SUMO Toolbox supports many other model types and hyperparameter optimization algorithms, yet these settings were used since experience showed they gave good results. A full discussion of the model types, optimization algorithms, genetic operators, etc. is out of scope for this chapter. Such details can be found in [273, 274].

In order to reliably gauge the quality of the models, a root relative square error (*RRSE*)

$$RRSE(\mathbf{y}, \tilde{\mathbf{y}}) = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \tag{5.10}$$

on a dense validation grid is calculated. $\mathbf{y}, \tilde{\mathbf{y}}, \bar{y}$ are the true, predicted, and mean true response values respectively. Intuitively the *RRSE* indicates how much better an approximation is than the most simple approximation possible (the mean) [25]. The size of this grid is $51^d, 15^d, 11^d, 7^d, 5^d$ for input dimension $d = 2,..,6$ respectively. Remark that we use a validation set since we want to accurately and objectively study the usefulness of ANN models. In general such a grid is not available and a cross validation measure (or similar) must be used.

Besides the *RRSE* we also recorded the Average Relative Error (*ARE*):

$$ARE(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \tilde{y}_i|}{|y_i|} \tag{5.11}$$

and the Maximum Relative Error (*MRE*):

$$MRE(\mathbf{y}, \tilde{\mathbf{y}}) = \max_{i=1..n} \left(\frac{|y_i - \tilde{y}_i|}{|y_i|} \right) \tag{5.12}$$

It is important to note that both the *MRE* and *ARE* are undefined for $y_i = 0$ and can be numerically unstable for y_i close to 0. Thus, in cases where the response value is small the *MRE* and *ARE* errors will be inflated. Care should thus be taken when interpreting the errors in the next section. The sampling settings were as follows: an initial optimized Latin hypercube design (constructed using the method described in [242]) of size 20 (10 in the 2D case) is used, augmented with the corner points. Each iteration a maximum of 50 (10 in the 2D case) new samples are selected using the LOLA-Voronoi active learning algorithm. The LOLA-Voronoi algorithm is a state-of-the-art sampling method described in [275] that works in both the $\mathbb{R}$ and $\mathbb{C}$ domains. Its strengths are that it scales well with the number of dimensions and makes no assumptions about the underlying problem or surrogate model type. At the same time it is able to automatically identify non-linear regions in the domain and sample these more densely. In this way the number of computationally expensive simulations can be minimized.

The termination criteria were determined by the application constraints and chosen as follows: a maximum of 1500 sample points or a *RRSE* validation error of 0.01.

Note that initially an error of 0.05 was deemed acceptable, however, in order to push the methods further we enforced a stricter requirement of 0.01. All 185 tests were run on the CalcUA cluster which consists of 256 Sun Fire V20z nodes (dual AMD Opteron with 4 or 8 GB RAM), running SUSE Linux, Matlab 7.4 R2007a, SUMO Toolbox v5.1 (settings not mentioned were kept to their defaults), administered by Sun Grid Engine (SGE).

5.4 Results

Table 5.1 summarizes the different results (the output ranges are given in table 5.2). For each output and each dimension the tables shows the errors of the model type that performed best (lowest value of *RRSE*). To help interpret the errors, the range of the output (determined by the validation grid) in each case is also listed. Figures 5.3, 5.4, 5.5, and 5.6 show how the accuracy scales in function of the number of samples and number of input variables.

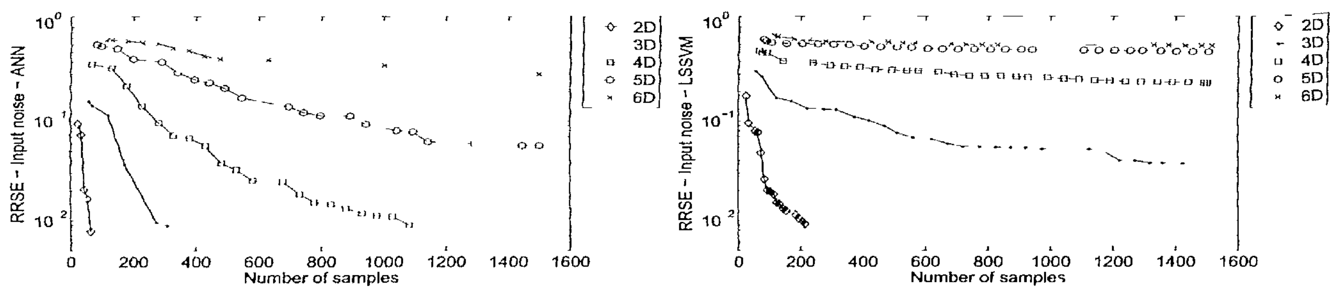


Figure 5.3: RRSE for the input noise-current $\sqrt{i_{in}^2}$ *(left: ANN, right: LS-SVM)*

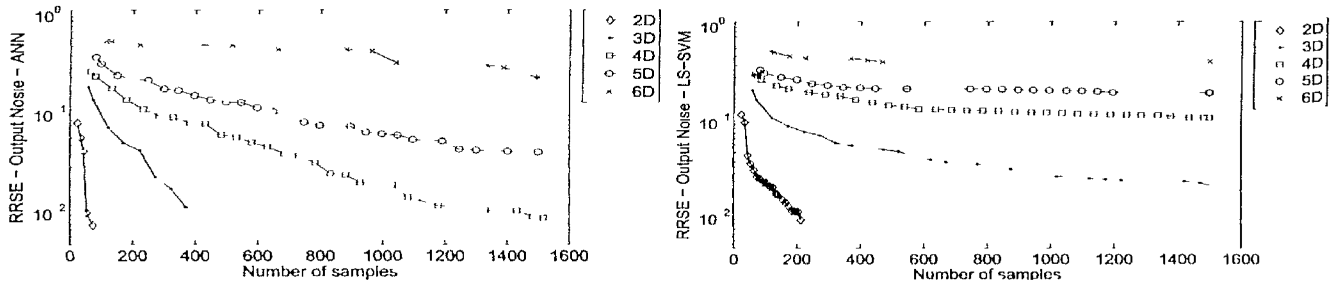


Figure 5.4: RRSE for the output noise-current $\sqrt{i_{out}^2}$ *(left: ANN, right: LS-SVM)*

From table 5.1 it is immediately clear that the ANN models perform best on all real valued outputs. For the 2D case the difference with Kriging, LS-SVM and rational functions, are minimal but for all the other dimensions the difference is easily 1 order of magnitude in favor of ANN. This is nicely illustrated in figure 5.4 (the plot for Kriging is not shown but is almost identical to the one for LS-SVM). The rather poor

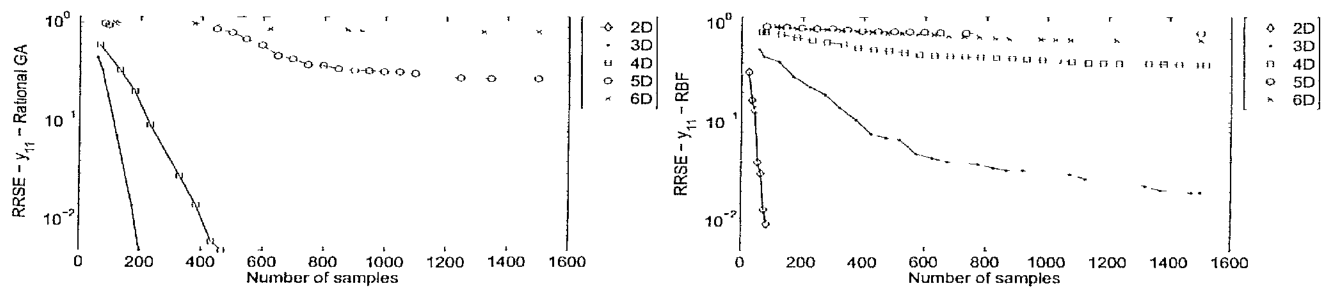


Figure 5.5: RRSE for y_{11} (left: Rational GA, right: RBF)

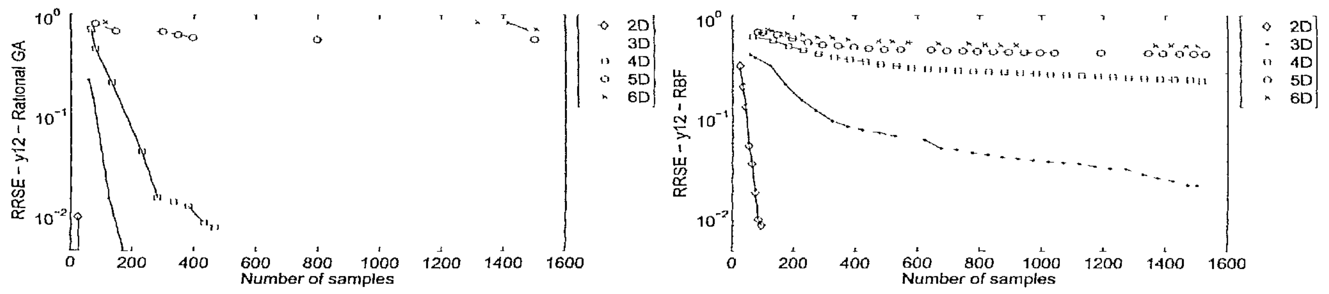


Figure 5.6: RRSE for y_{12} (left: Rational GA, right: RBF)

performance of Kriging is to be expected given the previous discussion in [142]. There is a close correspondence between SVM theory and Gaussian Process theory [127] yet results for LS-SVMs remain surprising given the many excellent results reported in literature (e.g., [168]). Nevertheless, these results are consistent with the authors' previous experiences on metamodeling problems with adaptive sampling. One would think the hyperparameter optimization algorithm would be to blame, yet previous tests with other algorithms (including Particle Swarm Optimization (PSO), grid search, pattern search and DIRECT), a brute force search through the hyperparameter space, and a comparison with manually determined settings by an expert give essentially the same results. In addition the intelligent restart strategy used by the SUMO Toolbox [142] adds an extra safety net to ensure that the model parameter space is searched thoroughly.

On the other hand the ANN models perform very well, failing to reach the 0.05 error target only in the y_{11}, y_{12} cases for 5-6D and the $\sqrt{i^2_{in}}, \sqrt{i^2_{out}}$ cases for 6D. For example, the error target is reached on $\sqrt{i^2_{out}}$ in the 5D case with roughly only 4.3 samples per dimension. For ρ_{real}, ρ_{imag} the same is true with only 3.3 samples per dimension. In addition the model complexity is quite small, starting from an initial complexity of 2 units in each of the 2 hidden layers, the complexity of the final models generated by the SUMO Toolbox did not exceed 500 weights. Together with the regularization this had the added benefit of keeping the responses smooth, capturing the global structure well (an important application requirement). A plot of the final ANN model of $\sqrt{i^2_{in}}$

for the 4D case is shown in figure 5.7. For comparison the corresponding LS-SVM model is shown in figure 5.8. The LS-SVM plot shows how the LS-SVM model is able to capture the rough structure of the response but fails to capture and track the 2 resonances along the $Lsn = -1$ axis (which move and grow sharper as the other parameters are varied). In addition, the LS-SVM models produce unwanted 'ripples' in some areas. In contrast, the ANN model is able to track the resonances quite well given the relatively sparse data distribution, while at the same time keeping the response smooth in the rest of the domain (thanks to the Bayesian regularization).

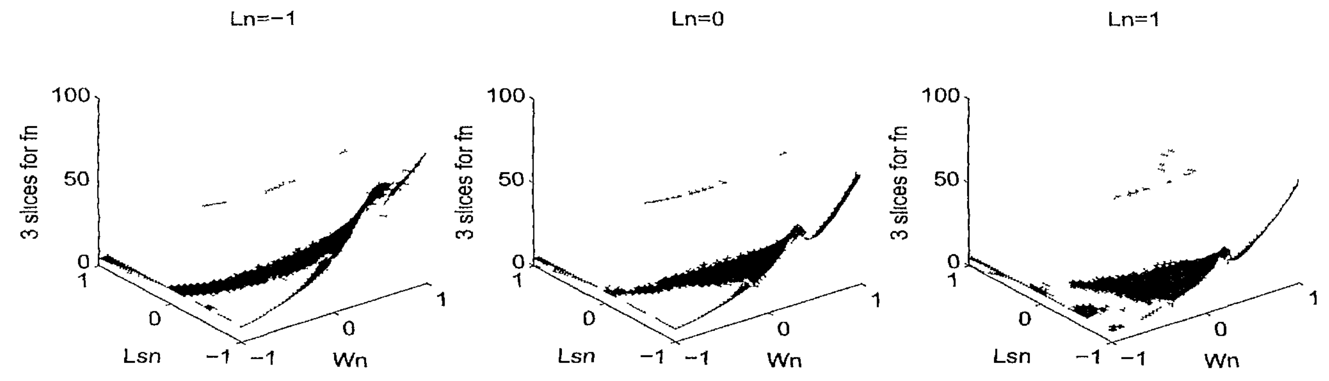


Figure 5.7. Final 4D ANN model for $\sqrt{i_{out}^2}$ (4-23-16-1 network, $L_{mn} = 0, V_{GTn} = 0, f_n \in \{-1,0,1\}$)

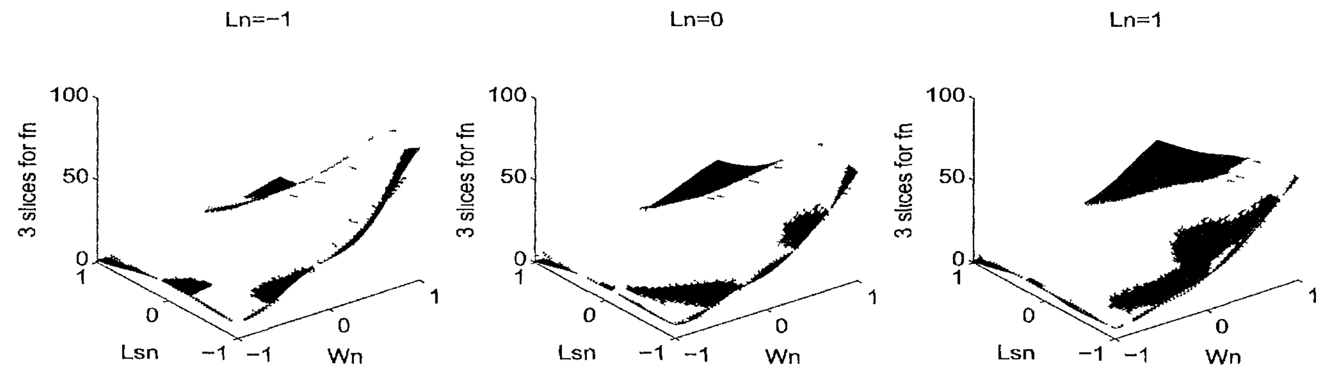


Figure 5.8. Final 4D LS-SVM model for $\sqrt{i_{out}^2}$ ($L_{mn} = 0, V_{GTn} = 0, f_n \in \{-1,0,1\}$)

For the admittances y_{11},y_{12} the accuracy of ANN models is roughly the same as the rational models, yet they require substantially more data points. However, this is only true up to 4 dimensions. For 5 and 6 dimensions, RBF models tend to do better than the rational functions (though the accuracy of the final models is still very poor), and the performance difference with ANNs is actually quite small.

The plots of the magnitude of y_{12} of the final rational and RBF models are shown in figures 5.9 and 5.10. Interestingly the structure of the plots is very similar to the plots for $\sqrt{i_{in}^2}$ and $\sqrt{i_{out}^2}$. Figure 5.9 shows a very good approximation, the response is smooth and shows a crisp resonance behavior. In contrast, the RBF model shown in figure 5.10 shows the same problems as the LS-SVM model in figure 5.8: the resonances are not captured sharply and the model suffers from unwanted 'ripples'.

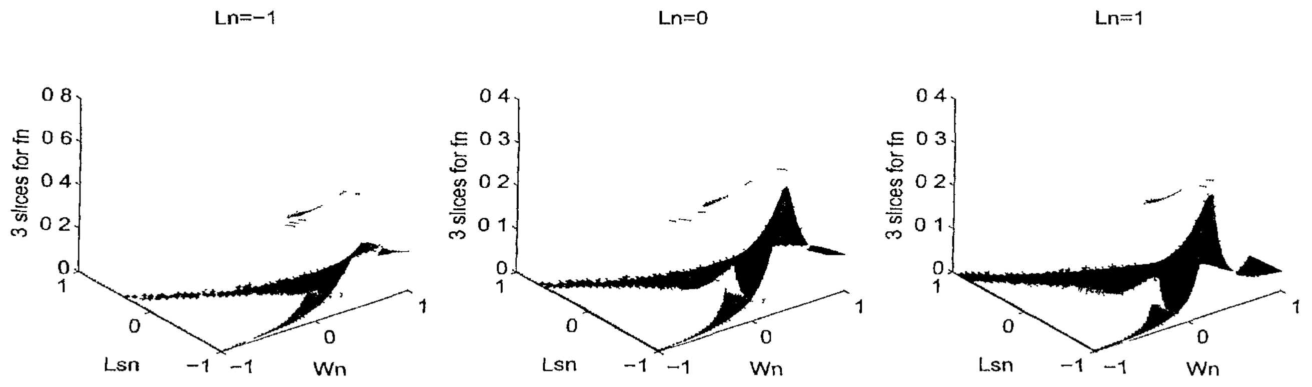


Figure 5.9: Final 4D rational model for $|y_{12}|$ ($L_{mn} = 0, V_{GTn} = 0, f_n \in \{-1, 0, 1\}$)

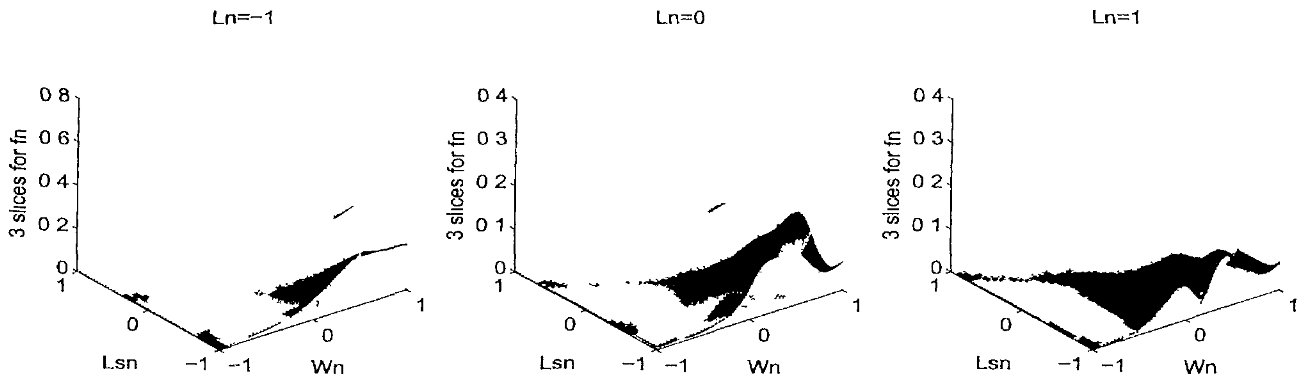


Figure 5.10: Final 4D RBF model for $|y_{12}|$ ($L_{mn} = 0, V_{GTn} = 0, f_n \in \{-1, 0, 1\}$)

Interestingly, the results for the correlation output ρ are somewhat different. There, modeling real and imaginary parts separately using ANNs clearly outperforms a direct approach using rational models. However, again for the 5D and 6D cases the results are much closer, though ANNs maintain a small margin. Good performance of the rational functions on the admittances is to be expected since y_{11} and y_{12} are themselves explicit rational functions. For ρ, the functions involved are not strictly polynomials so this could explain the discrepancy.

If we consider figure 5.6 the large gap between the 4D and 5D case for rational models is surprising. The authors suspect the poor scalability is due to the optimization algorithm (GA or HC) used to set the complexity (orders of polynomials, which variables belong in the denominator, etc.). As the dimensionality increases, there is less information per dimension yet more room for complex functions (the size of the search space increases exponentially). Thus the rational models generated by the algorithms involved quickly suffer from poles and have problems generating smooth responses. No significant difference was detected between the GA and HC algorithms, for this more runs would be necessary. Improving the algorithms to better combat overfitting should result in better scalability. However, the hyperparameter optimization algorithm is only part of the problem. As the dimensionality is increased, eventually the sample budget constraints will have to be relaxed if accurate global models are still required (asuming a non-trivial response to model). This flows directly from the curse of dimensionality and is true for all modeling approaches.

The algorithm for the RBF models is more robust but still performs very poorly. If we study the plots it seems the models have difficulty capturing the local non-linearities in the function. Roughly speaking the plots seem to indicate that, either the models are too smooth (thus missing important features) or they are too non-linear (capturing the features but failing to fit the smooth parts). This seems to be related to the problems the authors identified with Kriging models in [142] but a closer investigation is needed.

5.5 Conclusion

In this chapter we presented a comparison of the accuracy and scalability of different metamodeling methods on the complex task of modeling a whole LNA RF circuit block. ANNs have been found to perform very well overall, though rational functions should still be preferred if the underlying structure is rational and the dimensionality is low.

Some insight into the *accuracy - number of samples - dimensionality* relationship has been gained but more work remains. The ideal case would be to derive an empirical formula (for this and similar problems) that can easily be used to predict the number of data points (simulations) required to obtain a certain accuracy with a given model type. As a second step this formula should also take the required model complexity (i.e., number of weights in the case of ANN) into account. Such formulae (even if just a rough indication) would be extremely useful in the application of these methods in a real world, industrial setting.

In addition the possibility of including domain specific knowledge into the neural models needs to be studied (building upon the work by Zhang [74]). This should improve the results in higher dimensions.

		$y_{11-real}$	$y_{11-imag}$	y_{11}	$y_{12-real}$	$y_{12-imag}$	y_{12}	$\sqrt{\overline{i^2_{in}}}$	$\sqrt{\overline{i^2_{out}}}$	ρ_{real}	ρ_{imag}	ρ
2D	**Model**	**ANN**	**ANN**	**Rational HC**	**ANN**	**ANN**	**Rational GA**	**ANN**	**Rational HC**	**ANN**	**ANN**	**Rational GA**
	RRSE	8.755E-03	7.327E-03	5.714E-03	6.327E-03	4.045E-03	3.410E-03	8.758E-03	5.418E-03	3.730E-03	8.160E-03	8.849E-03
	ARE	9.317E-02	5.511E-01	1.825E-03	6.511E-01	3.820E-02	1.084E-03	1.395E-02	5.912E-03	1.567E-02	3.071E-02	8.738E-03
	MRE	5.605E+00	9.421E+02	1.441E-02	1.177E+03	1.169E+00	9.054E-03	2.087E-01	6.086E-02	2.368E+01	7.183E-01	3.164E-01
	# Samples	134	134	24	104	104	24	54	54	44	44	44
3D	**Model**	**ANN**	**ANN**	**Rational HC**	**ANN**	**ANN**	**Rational HC**	**ANN**	**ANN**	**ANN**	**ANN**	**Rational HC**
	RRSE	5.295E-03	2.981E-02	3.940E-03	8.252E-03	9.826E-03	1.747E-03	8.721E-03	9.414E-03	9.014E-03	9.612E-03	9.450E-03
	ARE	4.400E-01	4.612E-01	3.206E-03	2.317E-01	2.425E-01	2.148E-03	3.531E-02	6.670E-03	4.198E-02	2.091E+00	4.420E-03
	MRE	1.240E+02	1.294E-02	1.115E-01	1.548E+02	3.234E+01	2.923E-02	5.972E-01	1.364E-01	6.993E+00	5.397E+03	6.654E-01
	# Samples	1508	1508	158	658	658	208	308	508	308	258	1467
4D	**Model**	**ANN**	**ANN**	**Rational GA**	**ANN**	**ANN**	**Rational GA**	**ANN**	**ANN**	**ANN**	**ANN**	**Rational HC**
	RRSE	3.693E-02	5.971E-02	5.087E-03	3.040E-02	3.176E-02	8.600E-03	9.536E-03	9.172E-03	8.335E-03	8.878E-03	3.292E-01
	ARE	1.467E+01	1.770E-00	9.798E-03	6.336E-01	1.059E+00	8.651E-03	2.966E-02	6.346E-03	3.479E-02	1.534E+00	2.976E-01
	MRE	3.670E+03	8.722E+03	4.076E-01	1.773E+03	2.809E+02	2.047E-01	4.926E-01	5.995E-01	1.944E+01	1.243E+04	4.508E+00
	# Samples	1516	1516	466	1516	1516	466	916	1516	866	866	1516
5D	**Model**	**ANN**	**ANN**	**Rational GA**	**ANN**	**ANN**	**RBF GA**	**ANN**	**ANN**	**ANN**	**ANN**	**Rational GA**
	RRSE	9.954E-02	2.640E-01	2.741E-01	8.872E-02	7.881E-02	4.632E-01	5.711E-02	4.259E-02	2.547E-02	1.886E-02	4.122E-01
	ARE	2.495E+01	1.217E-01	3.508E-01	4.141E+00	9.901E+00	8.089E-01	2.075E-01	3.025E-02	2.606E-01	2.512E-01	3.988E-01
	MRE	8.553E+03	4.905E+04	1.179E+01	1.038E+04	5.591E+03	1.978E+01	2.176E+00	4.126E+00	2.942E+03	2.524E-02	1.129E+01
	# Samples	1531	1531	1532	1529	1529	1531	1530	1500	1532	1532	1532
6D	**Model**	**ANN**	**ANN**	**RBF GA**	**ANN**	**ANN**	**RBF GA**	**ANN**	**ANN**	**ANN**	**ANN**	**Rational HC**
	RRSE	2.454E-01	4.689E-01	6.711E-01	2.935E-01	3.049E-01	5.677E-01	4.189E-01	2.422E-01	3.907E-02	2.949E-02	5.740E-01
	ARE	1.323E-02	3.542E+00	8.642E-01	1.302E+00	1.773E+01	1.078E+00	9.649E-01	2.203E-01	4.696E-01	6.989E-01	5.251E-01
	MRE	2.334E+04	2.395E-03	2.015E+01	2.449E+02	1.485E+04	2.116E+01	3.338E+01	1.512E+01	6.925E+02	1.450E+03	1.536E-01
	# Samples	1509	1510	1511	1549	1503	1502	1511	1521	1514	1514	1514

Table 5.1: Final results for the best model type for each output

		$y_{11-real}$	$y_{11-imag}$	$y_{12-real}$	$y_{12-imag}$	$\sqrt{\overline{i_{in}^2}}$	$\sqrt{\overline{i_{out}^2}}$	ρ_{real}	ρ_{imag}
2D	**min**	6.98E-06	-1.31E-02	-1.45E-01	-2.89E-01	6.58E-01	1.67E+00	-9.87E-01	1.38E-03
	max	2.62E-02	1.30E-02	1.44E-01	-7.71E-05	1.06E+02	8.04E+01	9.45E-01	3.78E-01
3D	**min**	5.43E-08	-1.31E-02	-1.66E-01	-5.19E-01	5.77E-01	8.85E-01	-9.97E-01	-5 60E-05
	max	2.61E-02	1.31E-02	4.60E-01	-6.60E-06	1.10E+02	1.30E+02	9.67E-01	5.14E-01
4D	**min**	5.43E-08	-5.21E-02	-2.95E-01	-6.37E-01	4.08E-01	4.55E-01	-1.00E+00	-7.50E-01
	max	1.04E-01	5.23E-02	4.60E-01	-1.65E-06	2.00E+02	1.30E+02	9.88E-01	5.14E-01
5D	**min**	1.48E-08	-1.92E-01	-2.08E-01	-7.24E-01	4.08E-01	4.91E-01	-1.00E+00	-1.47E-02
	max	2.40E-01	1.86E-01	7.62E-01	-1.23E-07	5.85E+02	1.30E+02	9.94E-01	5.88E-01
6D	**min**	1.48E-08	-1.30E-01	-2.65E-01	-5.28E-01	4.08E-01	3.82E-01	-1.00E+00	-1.47E-02
	max	3.82E-01	1.75E-01	9.39E-01	-1 23E-07	4.88E+02	4.03E+02	9.95E-01	5.88E-01

Table 5.2: Response range for each output

Integrating Distributed Computing

I think there is a world market for about four or five electronic computers.
– Thomas Watson, IBM, 1943

6.1 Introduction

While the time needed for one evaluation of the original simulator is typically in the order of minutes, hours or even days, the surrogate function, due to its compact mathematical notation, can be evaluated in the order of milliseconds. However, the process of constructing accurate surrogates still requires evaluations of the original objective function. Therefore, if the process requires, say, 180 function evaluations and each evaluation takes 5 hours, the rate at which the design space can be explored is still relatively low. Such a simulation cost is of course unavoidable if a data based approximation method is used. The cost can be justified since (1) building a global surrogate is a one-time, up-front investment (assuming the problem stays the same), (2) adaptive modeling and adaptive sampling can drastically decrease the required number of data points to produce a good model and (3) distributed computing can reduce the "wall-clock" execution time by running simulations in parallel.

In this chapter we are particularly interested in the last point. The past two decades have seen the development of cheap Beowulf type clusters, computational grids, and virtualized clouds. The performance of which has come to rival classic state-of-the-art standalone supercomputers [276, 277]. With storage and computing power having become commodities it is only natural to consider how the surrogate modeling process can benefit from these evolutions. There are four main themes that underlie the motivations for this chapter:

1. improved model accuracy
2. reduction of the overall *wall-clock* time
3. increased data availability
4. increased interoperability

These four themes are closely related and can be mapped onto four concrete levels of integration: modeling level, resource level, scheduling level, and service level. The remainder of this chapter will discuss the necessity of these levels in more detail and provide two illustrative examples.

6.2 Distributed computing

While the developments that led to grid and cloud computing only span the last 20-30 years, the fundamental use case and need can be traced back to the early 1960s [278]. During that time computing and networking pioneer J.C.R. Licklider, originally an experimental psychologist at MIT, worked on psychoacoustics and was concerned with the amount of data he had to work with and the amount of time he required to organize and analyze his data. He developed a vision of networked computer systems that would be able to provide fast, automated support systems for human decision making [279]:

> *It will possibly turn out that only on rare occasions do most or all of the computers in the overall system operate together in an integrated network. It seems to me important, nevertheless, to develop a capability for integrated network operation [...] If such a network as I envisage nebulously could be brought into operation, we could have at least four large computers, perhaps six or eight small computers, and a great assortment of disc files and magnetic tape units – not to mention remote consoles and teletype stations – all churning away*

Licklider played an instrumental role in the development of ARPANET, which throughout the 1970s (with the development of ethernet) and 1980s led to the development of the Internet.

In parallel with the developments in networking and communication there was intense research on hardware and software applications for parallel computing. The focus was on algorithms, programs and architectures that efficiently enabled parallel execution within a local machine. Key developments from this time (late 1980s, early 1990s) include the Parallel Virtual Machine, Message Passing Interface (MPI), High Performance Fortran, and OpenMP [278]. However, as application developers began to develop large-scale codes that pushed against the resource limits of even the fastest parallel computers, some groups began looking at distribution beyond the boundaries of a single machine. Driven by large multi-disciplinary research challenges, the work on execution environments for parallel machines and distributed memory architectures evolved into the concept of gridcomputing which attained its peak at the turn of the century.

The grid grew from extending parallel computing paradigms from tightly coupled clusters to geographically distributed heterogeneous systems. At the same time acting as a platform for the integration of loosely coupled applications and for linking disparate resources (storage, computation, visualization, instruments). The first modern grid is generally considered to be the Information Wide-Area Year (I-WAY), developed as an experimental demonstration project for SC95. This led the way to popular infrastructure projects like Globus [280] and Legion [281] that explored approaches for providing basic system-level grid infrastructure. This infrastructure allowed resources to be coordinated to provide transparent, dependable, pervasive and consistent computing support to a wide range of applications. These applications can perform either distributed computing, high throughput computing, on-demand computing, data-intensive computing, collaborative computing or multimedia computing [282].

In recent years the term *cloud computing* has become dominant in the distributed computing literature. The definition given by Foster [283] is

> *Cloud computing is a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.*

Central to cloud computing is the concept of virtualization, abstracting the platform specific characteristics of a hardware architecture through a software virtual machine. Fundamentally, the idea is that a cloud user can configure a fully customized virtual machine and have it run and scale transparently and dynamically on a virtual 'cloud' of resources. There is no longer the limitation of one operating system having to correspond to one physical machine. Leading IT companies such as Amazon, Microsoft, Google and IBM, have announced huge investments into the development of cloud computing services. Both the cloud and grid computing paradigms promise to deliver computing resources as a utility, similar to traditional utilities such as water, electricity, gas and telephony. In this model, users have unlimited access to services, regardless of where they are hosted or how they are delivered (utility computing). According to Armbrust et al. [284], three aspects are new in cloud computing:

1. The illusion of infinite computing resources available on demand, thereby eliminating the need for cloud computing users to plan far ahead for provisioning. The complexity of capacity planning (the process of determining the capacity needed to meet changing demands) is therefore reduced significantly.

2. The elimination of an up-front commitment by cloud users, thereby allowing companies to start small and increase hardware resources only when there is an increase in their needs.

3. The ability to pay for use of computing resources on a short-term basis as needed (e.g., processors by the hour and storage by the day) and release them as needed, thereby rewarding conservation by letting machines and storage go when they are no longer useful.

Foster et al. [283] argue that cloud computing not only overlaps with grid computing, it has indeed evolved out of grid computing and relies on grid computing as its backbone and infrastructure support. The evolution has been a result of a shift in focus from an infrastructure that delivers storage and compute resources (such is the case in Grids) to one that is economy based aiming to deliver more abstract resources and services (such is the case in Clouds).

6.3 Modeling level

As stated in the introductary section, there are different ways concepts from distributed computing can be integrated into the surrogate model generation process. The first is on the model generation level.

The largest computational bottleneck in the surrogate modeling process usually is performing the necessary simulations. However, for relatively short simulations and expensive model types, the model generation cost can come to rival or even exceed the simulation cost. A typical example are neural networks, the use of backpropagation based fitting methods making them slow to train. This cost is amplified futher if an expensive (in terms of function evaluations) hyperparameter optimization algorithm is used, or a resampling based model selection procedure like cross validation. Given the availability of commodity multi-core and multi-CPU hardware it makes sense to leverage these resources and employ parallelism to reduce the model fitting cost. Parallelism may be applied within the model fitting routine itself (e.g., multi-threaded matrix inversion) or to fit multiple models simultaneously (e.g., during a population based hyperparameter optimization routine). Early work in this respect can be found in [285]. At this level there is usually no need to move outside the address space of the local machine since model training times are typically less than a minute. The network overhead would be too great. However, in cases where there is a lot of data, or the model fitting process is complex enough a distributed approach can be warranted.

Currently, the SUMO Toolbox allows for parallelization during the model building process in the following places:

- in any population based hyperparameter optimization method (GA, PSO, Differential Evolution, etc.)

- in k-fold cross validation
- during the management of the best model trace (cfr. section 4.5.4.5)
- when fitting an ensemble model
- within the implementation of some model types (e.g., Kriging)

Basically any place where more than one model needs to be fitted is a candidate for parallelization. Of course parallelization need not be restricted to models, complex sample selection algorithms may also benefit from parallelization.

In the SUMO Toolbox this kind of parallelization is made possible by the Matlab Parallel Computing Toolbox. If the local scheduler is available it allows for the parallel execution of Matlab code on a single machine. If also the distributed computing server is available, the same code can be transparently parallelized across a cluster, grid, or cloud of Matlab worker instances. This is illustrated in figure 6.1.

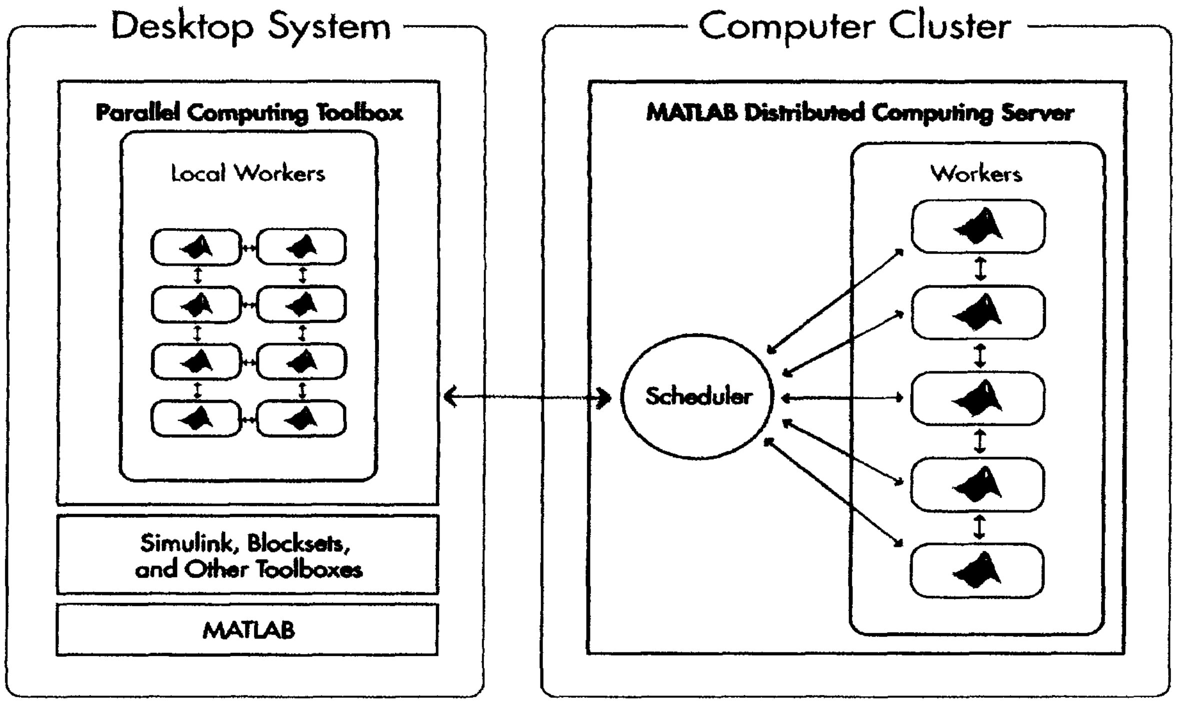

Desktop System
Parallel Computing Toolbox
Local Workers
Simulink, Blocksets, and Other Toolboxes
MATLAB
Computer Cluster
MATLAB Distributed Computing Server
Workers
Scheduler

Figure 6.1: Graphical illustrati on of the Matlab parallel computing capabilities (Source: http://www.mathworks.com)

6.4 Resource level

The most obvious and straightforward application of distributed systems to metamodeling is of course to parallelize the simulations that need to be performed. Given the SPMD (Single Program Multiple Data) nature of the problem this can be done quite easily. Many projects from surrogate based optimization have recognized this: Nimrod/O and Nimrod/G [286], the Visual Parametric Modeler developed as part of the Gridbus project [287], DAKOTA [88], GEODISE [288], the GIPSE toolset [289], ILab [290], and the work in [86,291–294].

Access to distributed resources is enabled through a software layer referred to as the middleware. The middleware is responsible for managing the resources (access control, job scheduling, resource registration and discovery, etc.), abstracting away the details and presenting the user with a consistent, virtual environment to work with. Major grid middlewares include Globus [280], Unicore [295], gLite [296], and BOINC [297].

In order to run different simulations in parallel, modeling software needs to transform requests to run a particular simulation into middleware specific jobs[1]. A graphical illustration of how the SUMO Toolbox solves this is shown in figure 6.2. The figure shows two ways to perform parallel simulation. If all simulations are done on a local multi-core or multi-CPU machine, the sample evaluator (cfr. section 4.5.3) maintains a pool of worker threads that enable multiple simulations to be performed simultaneously. Such local parallelization may be preferred for practical reasons like licensing or ease of use. However, for larger scale problems full distribution will be required.

In the distributed case, the workflow is a bit more complex. A *SampleEvaluator* subclass that supports distributed evaluation of the simulator code requires three subcomponents: a distributed backend, a poller, and a result processor object. A different grid middleware will require different implementations of these three components. To explain their interaction, let us take a concrete example involving the Sun Grid Engine (SGE) [298] middleware that is used in the applications in section 6.7. In our setup we use the CalcUA cluster at the University of Antwerp. This is a shared cluster, managed by SGE and accessible via a remote headnode. Job submission/querying/... is only possible from the headnode, to which one must connect using SSH.

Assume the toolbox is running on a local machine (not on the headnode). The flow of control is as follows (see figure 6.3): In step (1) the SUMO Toolbox control code passes the data points it needs simulated to the sample evaluator class. The sample evaluator wraps the passed points into generic *Job* objects, ensures all executables and dependencies are properly staged, and passes the *Job* objects to the distributed backend. The distributed backend then translates the generic *Job* object into SGE specific job submisison commands (i.e., *qsub* commands) in step (2). A *Poller* object is then started in step (3) that will continuously monitor the submitted jobs and detect when they finish (or fail). Since the cluster is only accessible through a remote headnode, steps 2-3 occur transparently through an SSH tunnel that is setup and maintained by the SUMO Toolbox. Once a Job has been completed, the poller notifies a *ResultProcessor* object

[1]Of course the simulation engine itself can be parallelized internally (e.g., through the use of MPI). This, however, goes beyond the scope of this chapter.

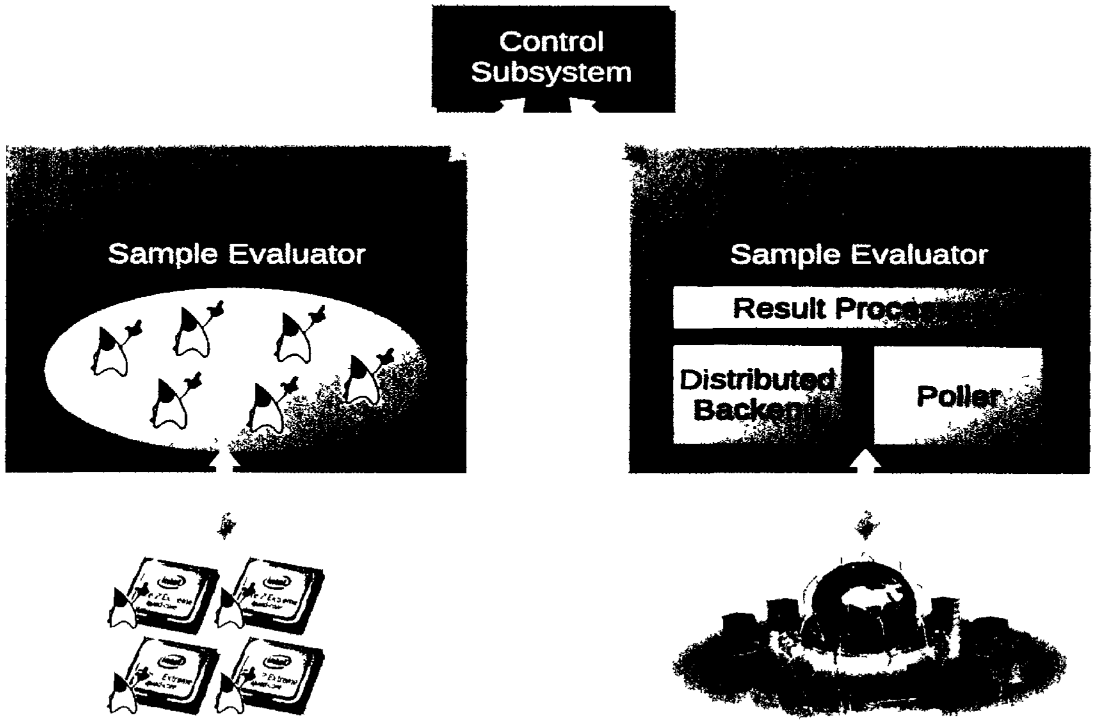


Figure 6.2: The SUMO Toolbox allows for the parallel execution of simulations. This may be done locally (on the same machine as the modeling) or on a cluster or grid of machines.

(step (4)) that retrieves the simulation results, does some sanity checking, and returns the completed results to the modeling code (through the output queue) in step (5). All this is of course done behind the scenes, the user need only provide some credentials and specify the right sample selector in the configuration file.

Thus, extending the SUMO Toolbox with a new grid or cloud computing sample evaluation backend (e.g., for the metascheduler GridWay [299]) can be done by providing a new {Distributed Backend, Poller, Result Processor} triplet.

As an aside, requests for data point evaluations only occur at the end of each modeling iteration. The model generation and selection subsystems run sequentially since Matlab only has a single thread of control. This means that the evaluation backend must be kept busy while the modeling loop is executed. Thus, in order to ensure full utilization of the distributed resources, the number of points requested for evaluation is not fixed but varies dynamically depending on the number of available nodes and the average time for one simulation (taken over a sliding window) and the duration of one modeling iteration.

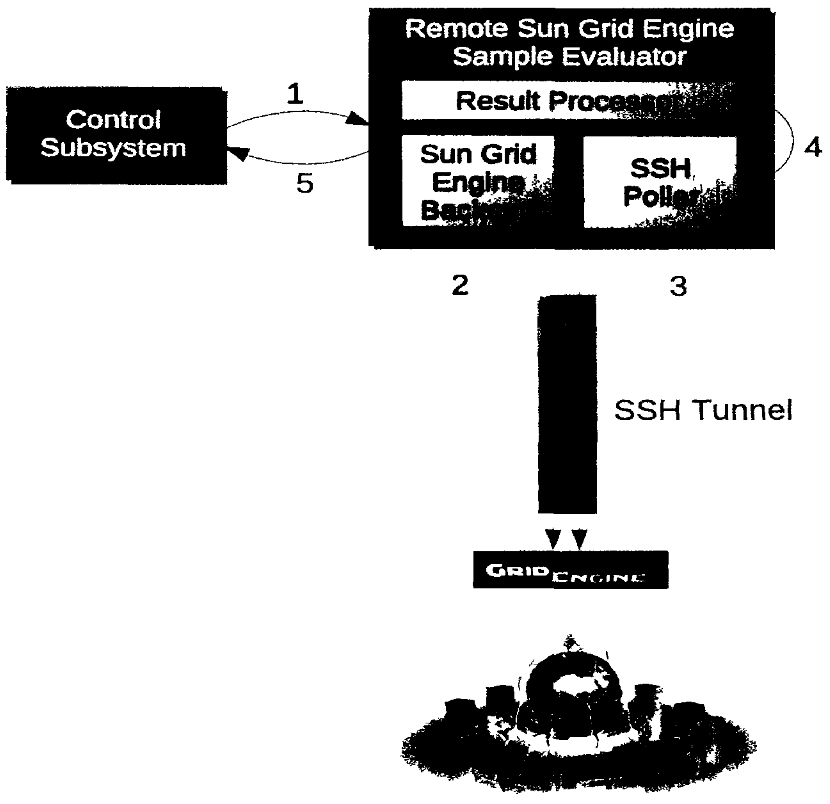


Figure 6.3: Evaluating data points (simulations) through a Sun Grid Engine administered cluster, accessible through a remote headnode.

6.5 Integration: Scheduling level

Simply running simulations in parallel already results in a significant decrease of the surrogate model generation time. Performance can be improved further if the intelligence of the sample evaluator (and associated queues) is increased. Remark that not all data points are equally important, a partial ordering exists. For example, data points lying close to interesting features (e.g., extrema, domain boundaries), or far from other data points should have a higher priority. These priorities are assigned by the sample selection algorithm(s) (cfr. section 4.5.2) and should be reflected in the scheduling decisions made by the sample evaluator and distributed backend. This type of integration is more difficult than the previous one since it requires a close interplay between the sample selection, model building and sample evaluation components. In particular it is important to keep the sample evaluation queue and the grid middleware queue filled, without overloading it with points that may become less interesting in the next modeling iteration.

The current version of the SUMO Toolbox uses a straightforward priority queue based on the sample point priorities in order to make scheduling decisions. This is already a good improvement but designing a good priority management algorithm that remains valid across multiple sampling iterations is still an open research problem. In addition, ideally, the sample evaluator should integrate seamlessly with the grid information system used (e.g., Ganglia [300]). Combining knowledge on queue availability, system load, disk usage, network traffic, ... with data point priorities would allow the sample evaluator to achieve an optimal $job - host$ mapping (i.e., the data points with the highest priority should be scheduled on the fastest nodes).

6.6 Integration: Service level

The previous three subsections exemplify the main reasons for traditionally turning to grid computing or distributed computing in general: computational power. However, the past few years Service Oriented Architectures (SOA) have become an increasingly popular (if not dominant) way to think about the grid or cloud. In this regard the distributed system is regarded as a heterogeneous collection of services, where each service provides access to a particular resource. Examples include services providing access to: a printer, a high performance numerical library, storage space, or CPU power. Users can connect to these services using standard technologies such as Apache River, Jolie [301], or SOAP and use them as part of complicated workflows. This is illustrated in figure 6.4.

In this sense, "automated construction of surrogate models" is a prime example of a service that a scientist or engineer can use to delegate surrogate model construction to. The advantages are obvious: there are no setup or maintenance costs, interfacing is straightforward, and horizontal scalability can easily be achieved. This allows for an easier integration of the surrogate model construction process into the larger design process, enhancing productivity.

A good example of such a SOA framework or Problem Solving Environment (PSE) [302] is the Geodise framework developed at the University of Southampton [4,288] or the problem solving environment discussed by Parmee et. al. [91]. The latter describes the initial development of the data modelling and search, exploration and optimisation processes of a Grid-enabled problem solving environment. This environment will enable a client to access coupled computational components sited at different "centers of expertise". Each center offers a data generation and analysis approach that aids a better understanding of the design domain whilst providing a route to the identification of appropriate high-performance design solutions. The intention is to support satisfactory, remote problem definition that leads to the selection and application of appropriate design search, exploration and optimization techniques. This should occur seamlessly so that the client is unaware that these processes are to be undertaken at different sites. Commercial tools that operate at this levels include ModelCenter and iSight.

Integration at this level is primarily a matter of implementation. At time of writing there is no readily available code that integrates the SUMO Toolbox in such a SOA.

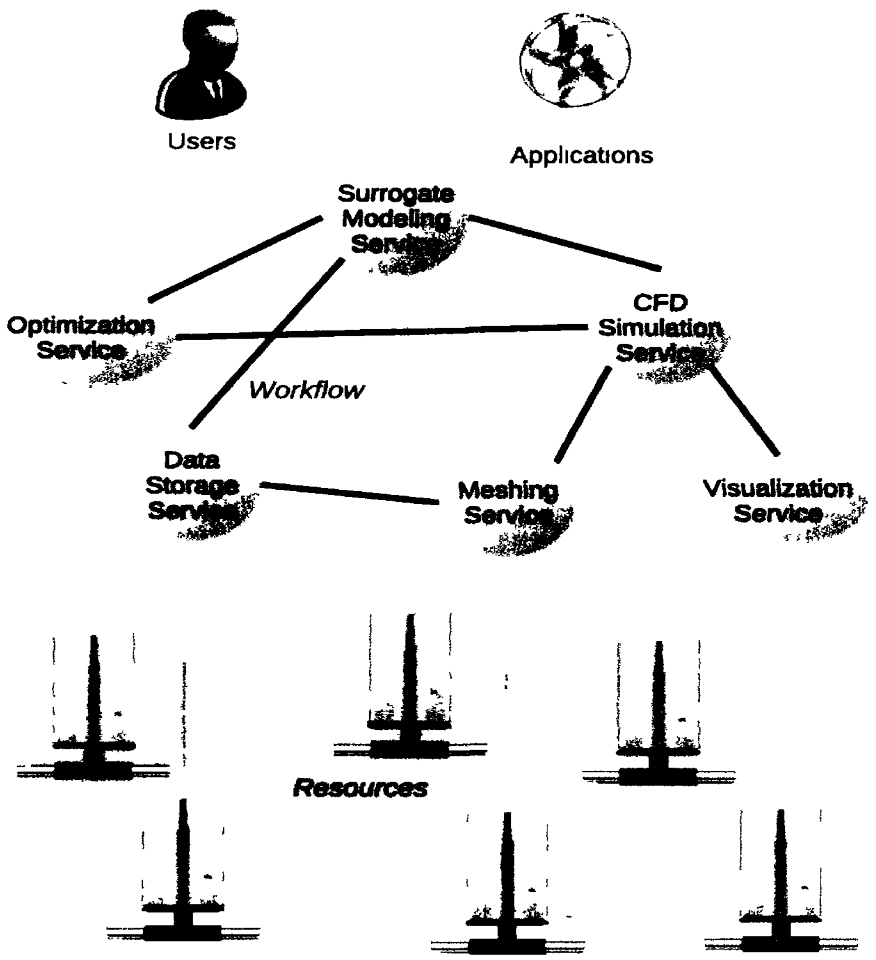


Figure 6.4: Service Oriented Architecture

But adding the necessary hooks (e.g., based on standard webservices, Apache River, or other enabling technology) to make this possible should not be difficult.

6.7 Applications

In this section we discuss two test problems to illustrate the discussions from the previous sections. We take a simple 2D analytical function, and a real world modeling problem from biophysics.

6.7.1 Analytical function

We use the *Academic2D* example problem from the SUMO Toolbox. It implements the following function defined on $[-1,1]^2$:

$$y = f(x_1, x_2) = \frac{e^{x_1+2}}{30 \cdot \Gamma(3x_2)} \tag{6.1}$$

A plot of the function is shown in figure 6.5.

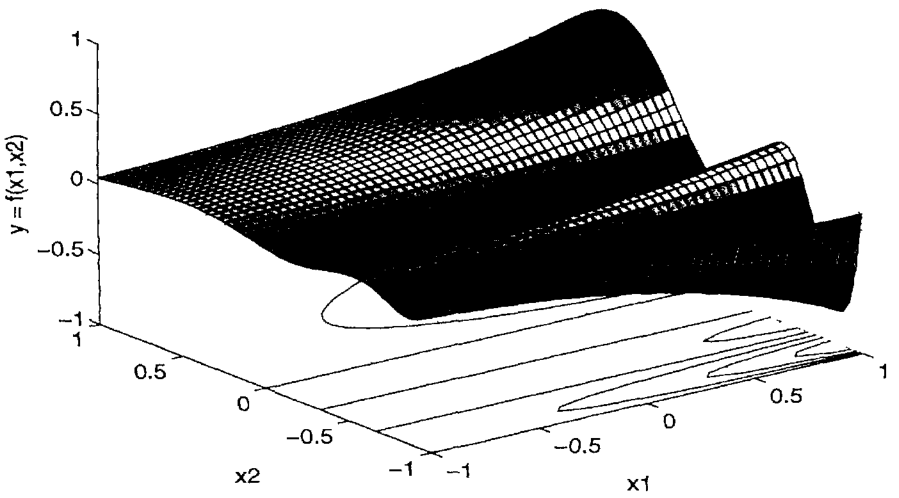


Figure 6.5: 2D analytical test function

6.7.1.1 Surrogate modeling

In this simple, illustrative example, the goal is to reproduce this analytical function with a minimum number of data points. As the model type we shall use ordinary Kriging with the (Gaussian) correlation parameters being determined through optimization of the likelihood. Internally the Kriging plugin also makes use of the model level parallelization features described in section 6.3. Since we are dealing with an artificial problem, we make the data more expensive by adding an artifical sleep of 10 minutes to each function evaluation. We know from experience that ordinary Kriging requires about 150 points to capture this function accurately. This means that traditional serial execution of the code will take about 25 hours.

To reduce this cost we illustrate the use of the SGE based sample evaluator described in section 6.2 and illustrated in figure 6.3. From a user point of view this simply means setting the correct *id* in the toolbox configuration file and setting the *username*, *host URL*, and *remote working directory* options. Authentication, job submssion, monitoring and retieval happens transparently. In our setup we use the CalcUA cluster at the University of Antwerp. This is a shared cluster of 256 Sun Fire V20z nodes (dual AMD Opteron 250, 2.4 GHz). To make testing easier we use the fast queue consisting of 20 nodes instead of the (heavily loaded) main queue of 230 nodes.

All other options are kept to their default values (as defined in v6.2.1). Higher priority samples (as determined by the sample selection algorithm) will be scheduled first.

6.7.1.2 Results

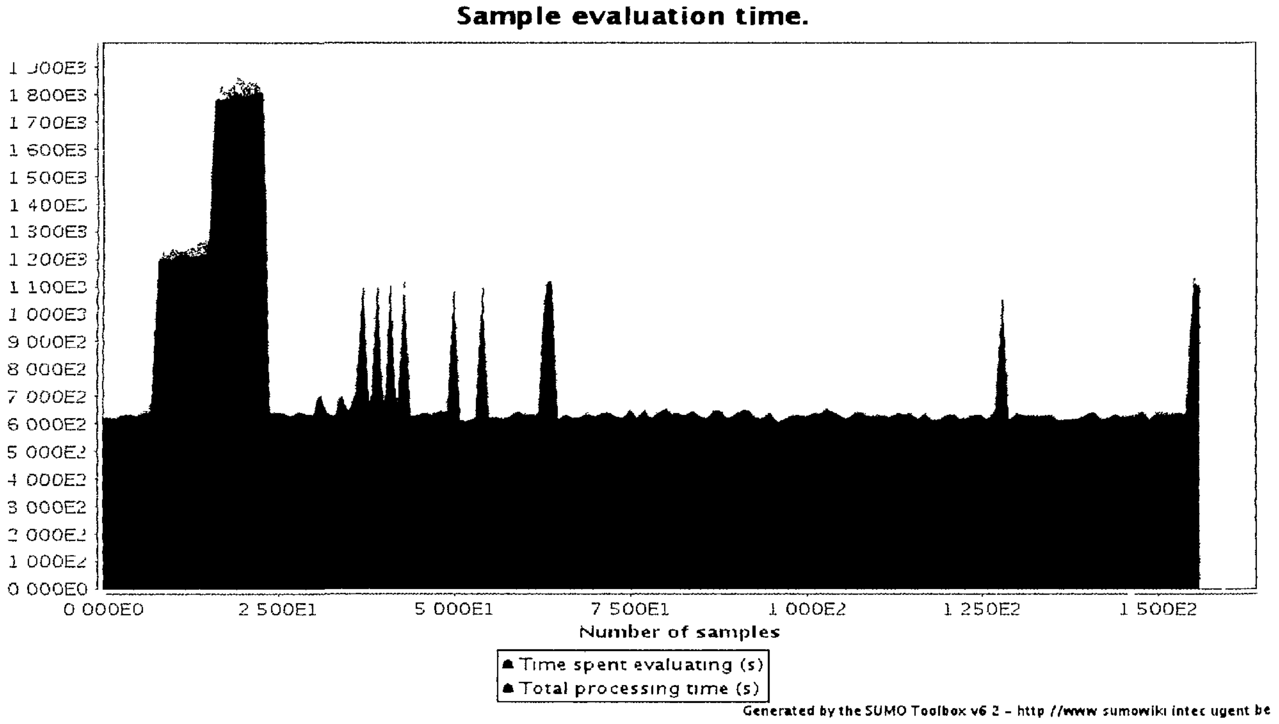


Figure 6.6: SUMO profiler showing the simulation time for each sample point (analytical example). The spikes are due to points having to wait in the SGE queue before being scheduled.

The evaluation time for each point is shown in in figure 6.6. Figure 6.7 shows the elapsed wall clock time during the modeling process. From this we see that the total time is 150 minutes or 2.5 hours. Compared to pure serial execution, this is effectively a 10-fold speedup. The 10-fold speedup can also be seen from the node utilization profiler which tracks how many points are running concurrently (figure 6.8).

6.7.2 Biophysical Application

This application comes from biophysics and concerns the modeling of the tympanic membrane (eardrum) in the human ear. Its place in the overal anatomy of the human ear is shown in figure 6.9. This section draws from [303] where a detailed overview of this application is given.

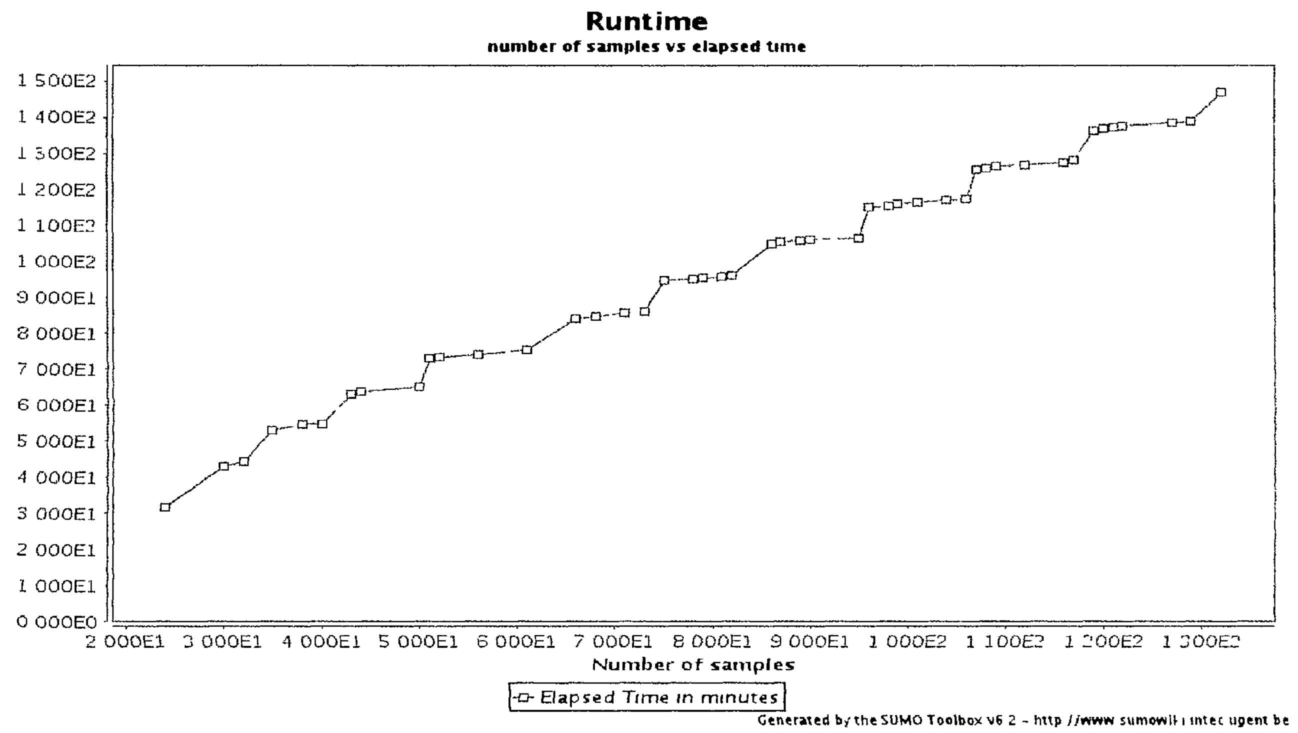


Figure 6.7: SUMO profiler showing the elapsed wall clock time for the analytical example. The toolbox terminates after 2.5 hours, 10x faster than pure sequential execution.

6.7.2.1 Background

Correct quantitative parameters to describe tympanic membrane elasticity are an important input for realistic modeling of middle ear mechanics. However, so far these have not been determined accurately. Current finite element models are mainly restricted to acoustical sound pressures and low acoustical frequencies, and good data for the mechanical properties of the tympanic membrane are still lacking. It is known, however, that tympanic membrane elasticity has a significant influence on the resulting output [303].

Thus a setup was developed to determine tympanic membrane elasticity in situ. The measurement method consists of doing a point indentation perpendicular on the membrane surface; measuring the indentation depth, resulting force and three-dimensional shape data; simulating the experiment with a finite element model and adapting the model to fit the measurements using optimization procedures.

The tympanic membrane sample (in this case obtained from a rabbit) was placed on a translation and rotation stage, a schematic drawing is shown in figure 6.10. Indentations in and out in a direction perpendicular to the surface membrane were carried out using a stepper motor with indentation depths up to 2 mm. The resulting force was measured with a load cell and the exact indentation depth was assessed with a Linear Variable Differential Transformer (LVDT).

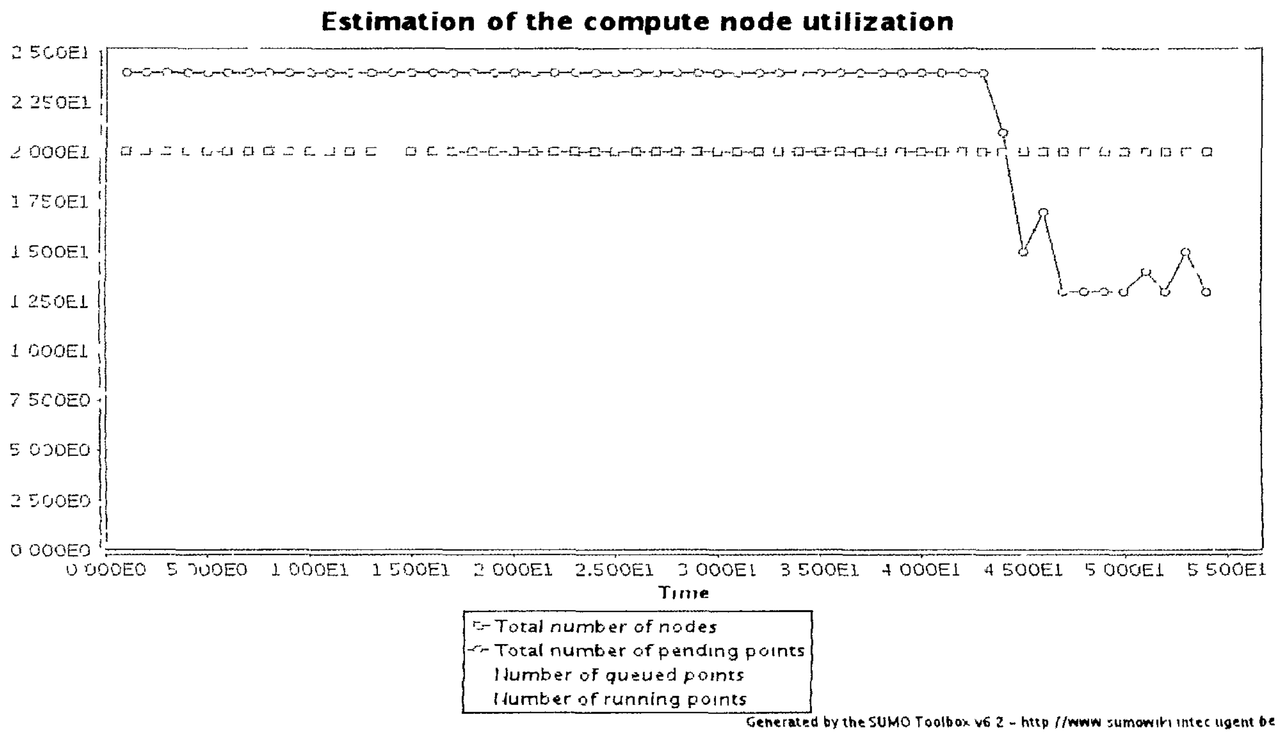


Figure 6.8: SUMO profiler showing the estimated utilization of the shared compute nodes during the modeling of the synthetic function.

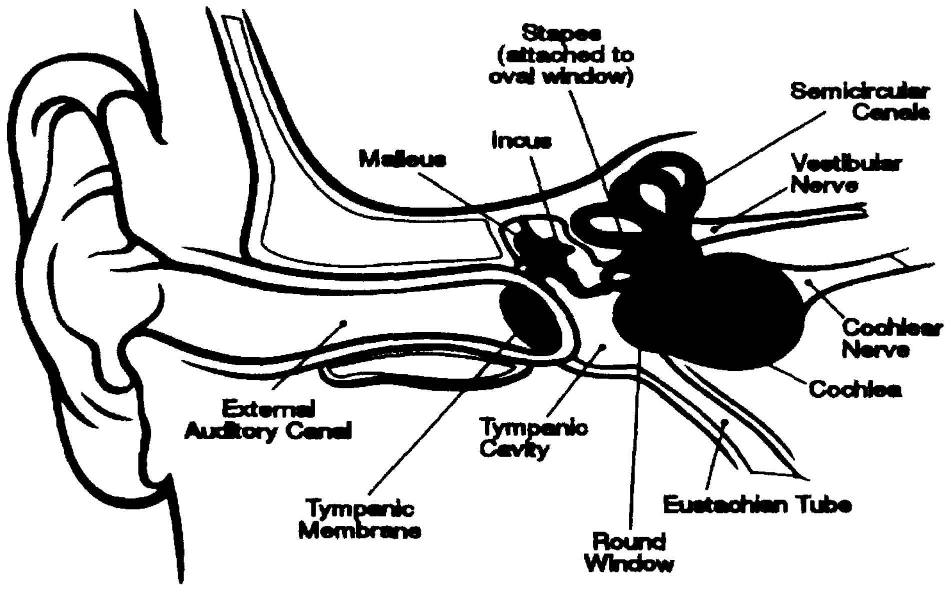


Figure 6.9: Anatomy of the human ear (Source: Wikipedia)

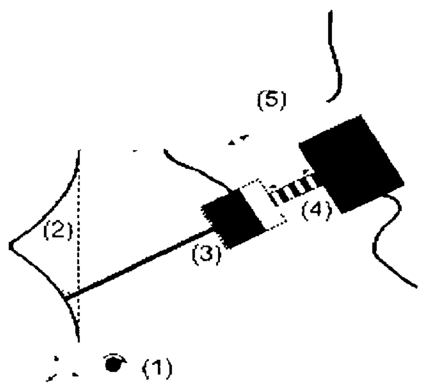
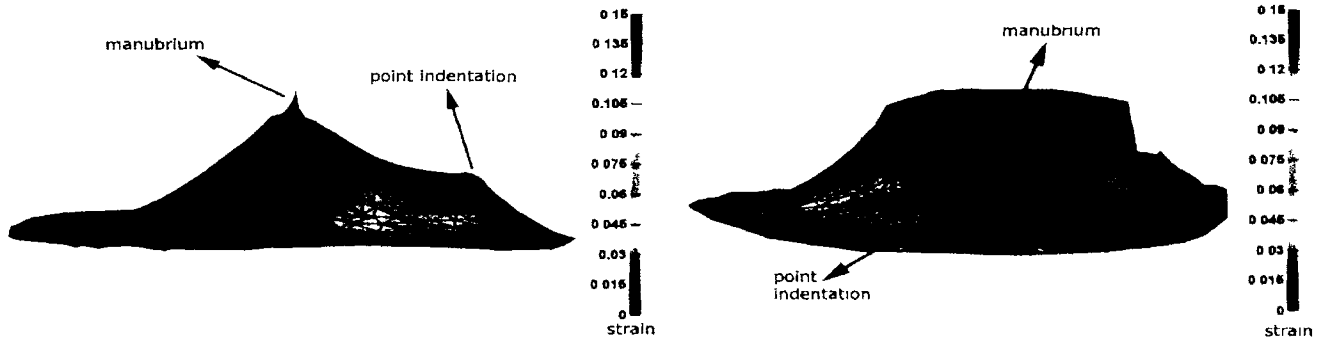


Figure 6.10: Schematic drawing of the point indentation setup: (1) translation and rotation stage, (2) tympanic membrane sample, (3) needle connected to a load cell, (4) stepper motor and (5) Linear Variable Differential Transformer.

In order to construct a model, an LCD-Moiré profilometer was used to obtain the three-dimensional shape of the membrane before and during indentation. On the basis of these Moiré shape images a highly detailed non-uniform finite element mesh was created. In the needle indentation area and in the manubrium neighbourhood, mesh density was increased. This is illustrated in figure 6.11. The tympanic membrane was modeled as a linear isotropic homogeneous elastic material which is described with two independent elastic parameters: the Young's modulus E and Poisson's ratio ν. The numerical simulations were performed with the finite element code FEBio, which is specifically designed for biomechanical applications.



(a) Posterior view (b) Superior view

Figure 6.11: Finite element model of the tympanic membrane with indentation. The number of membrane shell elements equal to 5988. The effective strain in the point indentation area after indentation rises up to approximately 15%.

Determining the value of the linear elasticity parameters is done by minimizing the discrepancy between the model and the experimental measurements. Namely, by calculating

$$\arg\min_{E,m}(error_{force}) \quad \text{with} \tag{6.2}$$

$$error_{force} = \frac{1}{N}\sum_{j=1}^{N}(F_{exp}(q_j) - F_{mod}(q_j))^2 \tag{6.3}$$

N is the number of measured points, q_j the indentation depth, $F_{exp}(q_j)$ the experimental force and $F_{mod}(q_j)$ the simulated force.

6.7.2.2 Surrogate modeling

In [303] the infill infrastructure of the SUMO Toolbox was used to optimize $error_{force}$. In this chaper we are now interested in capturing the full 2D landscape defined by $error_{force}$. Again we use the ordinary Kriging plugin for this purpose. The added difficulty here was that sometimes the FEBio code did not converge properly, leading to a noisy surface with potential outliers. For this reason the Kriging plugin was also configured to optimize a parameter λ together with the correlation parameters. λ controls the degree of interpolation of the Kriging model, with $\lambda = 0$ meaning exact interpolation. All other settings were again kept to their defaults, with only the initial design (an optimal Latin Hypercube) reduced to 5 points instead of 25.

6.7.2.3 Results

A plot of the final model of $error_{force}$ after 920 samples is shown in figure 6.12. Figures 6.13, 6.14, and 6.15 show the sample evaluation time, estimated speedup over serial execution, and node utilization respectively. The average simulation time is about 25 minutes (remember that this includes internal SGE queueing time). As can be seen from figure 6.15, the toolbox will always try to maximize the amount of nodes used. At the same time taking care not to have too many waiting or pending points, since once they are submitted they can no longer be replaced with potentially more useful points. The resource usage will expand or contract depending on node availabilities. The speedup over serial execution varies from time to time, ranging from about 3 to a peak of 11.

6.8 Conclusion

There are various levels at which the surrogate modeling process can benefit from distributed resources and serive oriented architectures. Most of the work so far has been done on the most obvious level of integration: parallization of the simulations themselves. The challenge here is interfacing with the different middlewares in a flexible, extensible manner. Each middleware has its own characteristics and semantics which

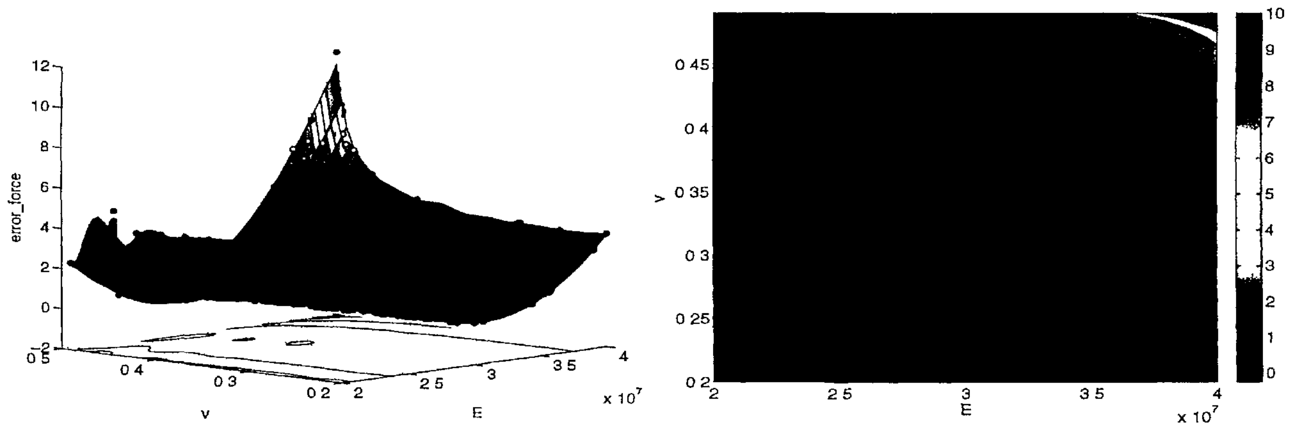


Figure 6.12: Kriging surrogate of $error_{force}$.

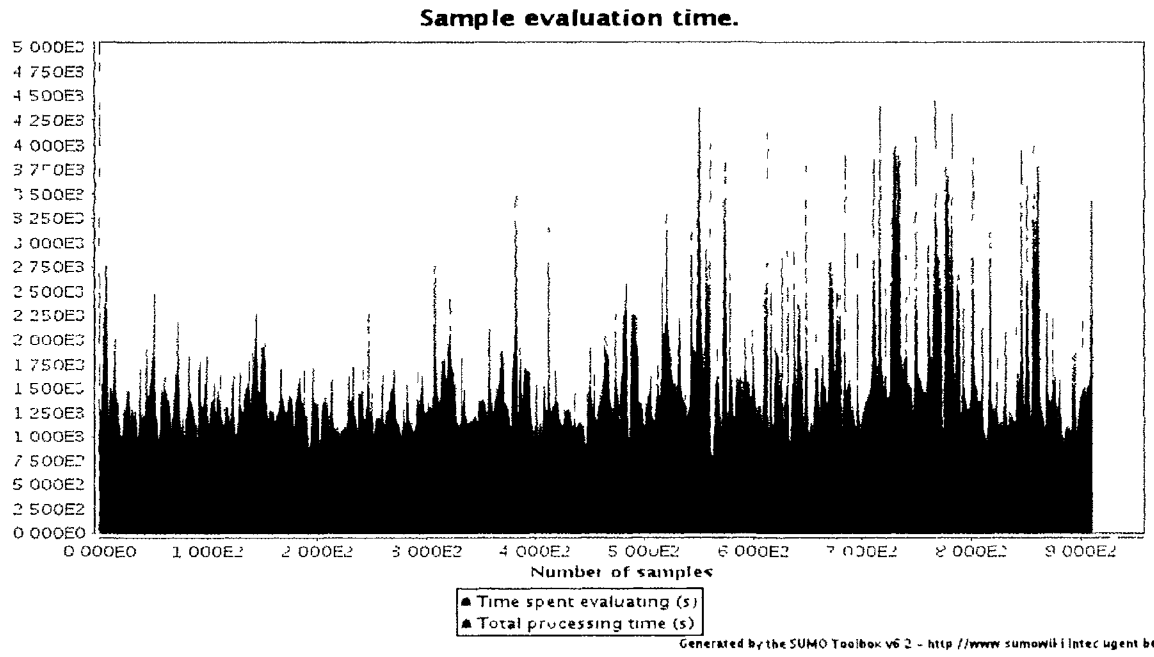


Figure 6.13: SUMO profiler showing the simulation time for each sample point (biophysics example). Note that this time also includes time spent in the SGE queue.

makes it difficult to support different middlewares in a transparent way. Luckily work on different meta schedulers (e.g., GridWay, ProActive) and standardization efforts (CoG kit, DRMAA, ...) are underway to tackle this problem.

Less work has been done on the higher levels of integration. In particular there is still room for improving the priority management policy and integrating the resource information system with the scheduler. Integration at the service level by exposing functionality through well defined web service APIs is also a topic of further work.

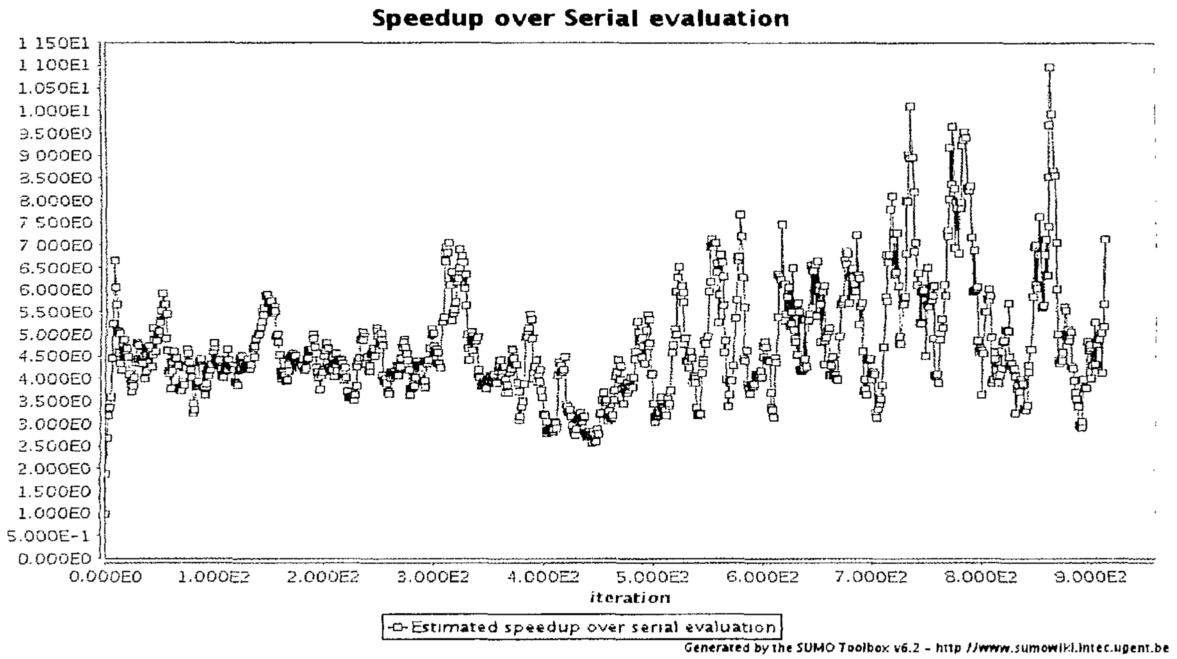


Figure 6.14: SUMO profiler showing the estimated speedup over serial executionfor the biophysics example.

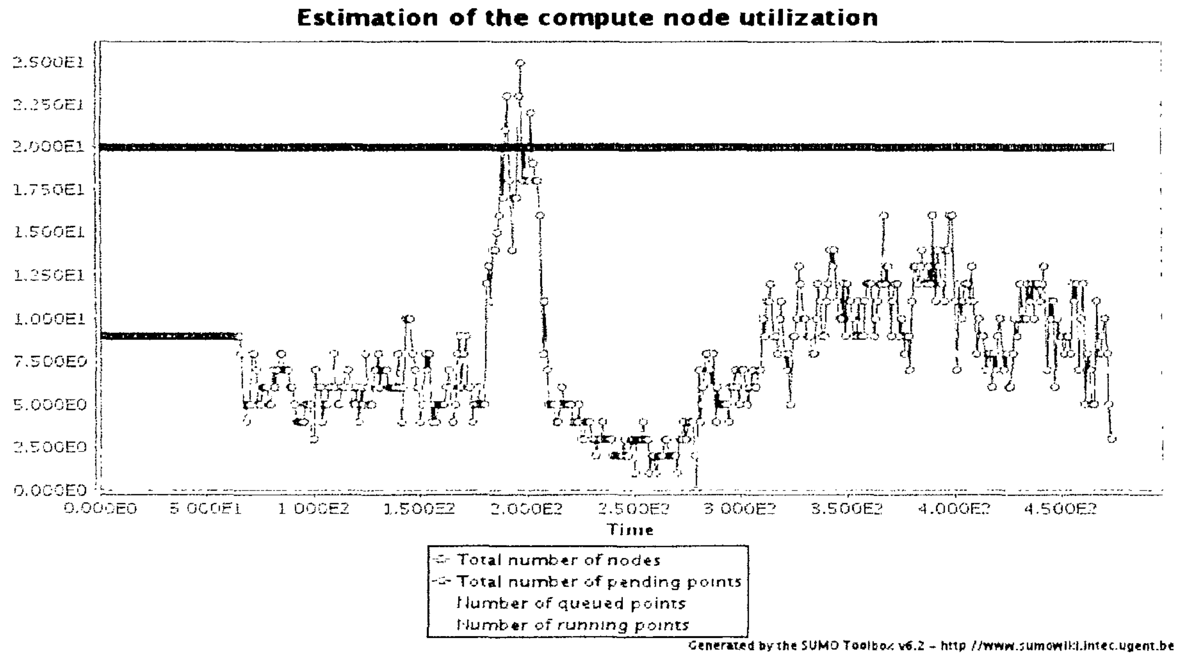


Figure 6.15: SUMO profiler showing the estimated utilization of the shared compute nodes during the modeling of the biophysical example.

7

Evolutionary Model Type Selection

It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change.

– Charles Darwin

Recall from the discussions in chapters 2 and 3 that the primary users of global surrogate modeling methods are domain experts and engineers. Few of these will be experts in the intricacies of efficient sampling and modeling strategies. Their primary concern is obtaining an accurate replacement metamodel for their problem as fast as possible and with minimal overhead. The details involved in model selection, model parameter optimization, sampling strategy, etc. are of lesser or no interest to them. Thus any automation that can be introduced into the surrogate modeling process would be very helpful. In this spirit, this chapter explores an automated way to help answer the always recurring question from domain experts *"Which approximation method is best for my data?"*. An evolutionary algorithm is presented that combines automatic model *type* selection, automatic model parameter optimization, and sequential design exploration.

7.1 Background

Arguably the hardest problems in science and engineering are those that involve mimicking or understanding biological systems: auto-correction during DNA transcription, self organization, language processing, vision and cognitive reasoning. The complexity of such systems is staggering, yet at the same time their implementation has often turned out to be breathtakingly elegant. Confucius' dictum rightfully comes to mind: *A common man marvels at uncommon things; a wise man marvels at the commonplace*". The mechanism nature has used to achieve this is evolution through natural selection, a theory proposed and formalized by English naturalist and meticulous experimentor Charles Darwin in his 1859 seminal publication *On the Origins of Species*. While Darwin is rightfully accredited with laying the foundations of modern evolutionary biology and invalidating the contemporary theories on the hierarchical nature of species organization, various evolutionary ideas had already been proposed at the time. The most well known example being the work by Alfred Russel Wallace. The theory on adaptation through natural selection seems conceptually very simple, yet the mental leap required to fathom its far reaching implications is considerable. Contemplating how deep its influence reaches out, it actually becomes hard to fault Darwins' Victorian contemporaries for their outrage. It took until the 1930s and 1940s for its full significance to be accepted, helped by the re-discovery of the work by Mendel in 1900.

Inspired by the complexity of the problems evolution through natural selection seems to have solved, Man has ever since attempted to (crudely) replicate this process in the lab and virtually on computers. Though the first personal computer (the IBM 5150PC) did not appear until 1981 and the computing paradigm was still very new, the Norwegian scientist Nils Aall Barricelli was already applying it to to the simulation of evolution in 1954. Though many researchers picked-up on the idea, it wasn't until work in the early 1970s, by researchers at the University of Michigan led by John Holland, that the use of evolution as a problems solving method became widely recognized. Since then, evolutionary algorithms (EA) have been widely used in many diverse domains with applications ranging from language processing and artificial life, to hull optimization in aerospace.

Within this chapter we build on this work and apply some of the concepts involved to the problem of surrogate model type selection.

7.2 Biological Foundations

Since EAs were inspired by processes in nature, appreciation of these mechanisms is necessary. Given a population of organisms, evolution can occur only if the following two conditions are satisfied:

1. In the population there must exist variation for some trait and this variation must be heritable. Examples are beak size, skin complexity, eye color, etc.
2. There must be differential survival and reproduction associated with the possession of that trait.

The first condition requires each trait to have a genetic basis (genotype), that this genotype varies between individuals, and that it can be passed onto offspring. The second condition states that the genetically encoded trait must have a phenotypic expression that incurs some advantage or disadvantage for survival and reproduction in an environment. Together these conditions form the basis for adaptation through Natural Selection:

> *I have called this principle, by which each slight variation, if useful, is preserved, by the term Natural Selection.*
>
> - Charles Darwin, The Origin of Species

7.2.1 Heritable variation

It cannot be stressed enough that variation must be heritable (the first condition for evolution). Each phenotypic effect must have a matching code in the organism's genome, and this code can be passed onto future generations. Traits with no genetic basis at all (acquired traits) *play no role* in evolution since they have no effect on offspring and are thus not susceptible to selection. The theory where acquired traits *do* play a role in evolution is known as Lamarckian evolution[1] (versus Darwinian evolution) but no evidence for this theory has ever been found and the scientific consensus is that there never will be[2].

While this genetic aspect may be obvious in retrospect, in Darwin's time it posed a great problem since he, unaware of Mendel's work in 1865, could not explain how traits were inherited or blended. It wasn't until Mendel's work on genetics was rediscovered in 1900 and reconciled with Darwinian evolution in 1930 that the foundations for the current theory *modern evolutionary synthesis,* or *neo-Darwinism* were laid.

Of course, simply requiring that traits have a heritable, genetic basis is not enough to explain the enormous diversity of species that exist in the world. If each parent would simply pass on an exact copy of its genes, there never would be any new variation, the gene pool would remain static and no new species would arise. Therefore, as genetic information is passed onto offspring it undergoes mutation and, in some cases, recombination.

7.2.1.1 Mutation

Mutation is a change in the genetic information and can be caused by:

- copying errors in the genetic material during cell division
- exposure to ultraviolet or ionizing radiation

[1] As a historical sidenote, the inheritance of acquired characteristics is not the aspect of his theory that Lamarck himself emphasized, he simply took over the conventional wisdom of his time and grafted to it other principles like 'striving' and 'use and disuse' [304].

[2] This, however, has never stopped its ideas being applied to solve engineering problems (e.g., Memetic optimization [305])

- chemical mutagens or viruses
- deliberately under cellular control during processes such as meiosis or hypermutation

Mutation can occur in any bodily (somatic) cell, but it only contributes to evolution if it occurs in the germ cells responsible for reproduction (we only consider multi-cellular organisms, excluding plants). Such mutations are called germline mutations (versus somatic mutations). Mutation is a mechanism by which *new* information can be added to the gene pool. Figures 7.1 illustrates some examples of mutation.

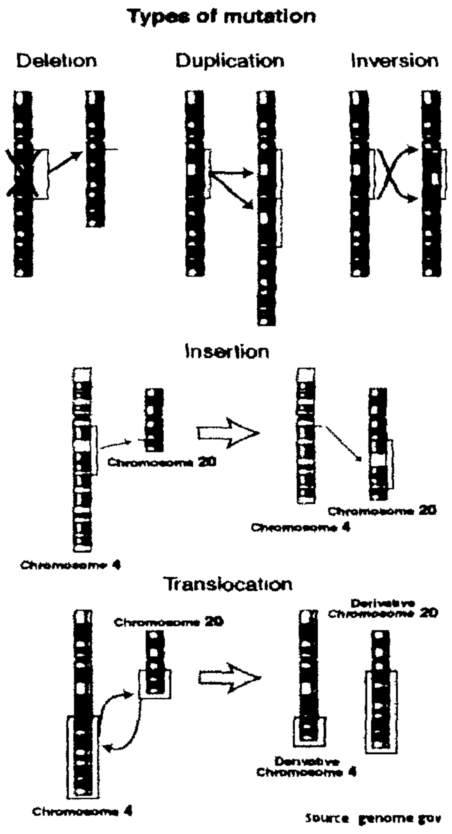


Figure 7.1: Examples of mutation

7.2.1.2 Recombination

The second genetic operator is recombination, also referred to as crossover. It is only required for sexual reproduction (versus asexual). In humans, recombination occurs during meiosis, the process during which a diploid cell (a cell containing the full 46 chromosomes, 23 from each parent) divides into four haploid cells, each containing only 23 chromosomes. During sex, some of these haploid cells will then recombine with haploid cells of the other sex to form a zygote, and eventually, a new individual.

Recombination occurs during the Prophase I of meiosis, the chromosomes from each parent pair up and randomly exchange information through crossing over. The 23 pairs then break-up again and the 46 chromosomes divide themselves randomly (but equally) over two new haploid cells, each containing only 23 chromosomes. These cells are then duplicated in a way similar to somatic cell division (mitosis) to form the final four haploid cells. An important feature of these four cells is that the combination of genes they carry on their 23 chromosomes is a unique mix of the genes present in the original single cell.

Thus recombination is a mechanism that combines *existing* variation in the gene pool but cannot create new variation. Crossover is illustrated in figure 7.2.

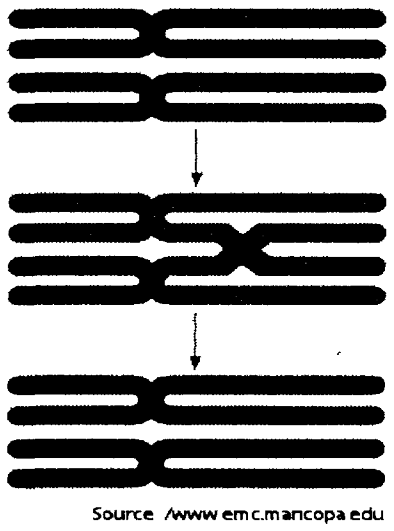


Figure 7.2: Example of crossover

7.2.2 Differential survival and reproduction

Having heritable variation available for a trait is only one side of the equation. It is how the expression of this trait helps the organism gain reproductive success that drives evolution. The advantage of having a particular trait is completely determined by the environment (climate, geography, predators, available resources, number and species of other organisms, etc.) the organism finds itself in. Traits that make the organism better adapted to its environment, i.e., increase the chances of successful reproduction, will have a larger probability of being passed on to the next generation.

The classical example is that of the *Peppered Moth* in England. Prior to 1800, the moth typically had a light pattern which camouflaged it against the light tree trunks and lichens they rested upon. With the advent of the industrial revolution, however, soot and other industrial waste darkened the trees and killed off lichens. The lightly colored moths had suddenly become more visible to predators, decreasing their chances of surviving until reproductive age. In contrast, the darker colored moths suddenly found themselves more camouflaged and as a result their percentage of the population increased.

When discussing such examples it is common to use the following terminology: "Natural selection selected *against* light color", "There is positive selective pressure *for* dark wings", etc.. To the uninformed reader this may seem like Natural Selection is a directed process, working towards some unidentified goal. Nothing could be further from the truth. At its core, evolution through natural selection is a stochastic process with no intrinsic direction or preference whatsoever. This sense of undirectedness implies that the solutions found by natural selection are by no means guaranteed to be optimal in any way. Or as Darwin, so vividly put it in a letter to his friend Joseph Hooker:

> *"What a book a devil's chaplain might write on the clumsy, wasteful, blundering, low, and horribly cruel works of nature!"*

7.3 Evolutionary Algorithms

7.3.1 History

Given the success of evolution by natural selection, scientists were quick to try to replicate this success for man-made problems. The use of EAs concentrated on two domains: modeling and validation of biological evolution, and global search. This thesis is only concerned with the latter.

The computer simulation of evolution dates back to the early 1950's by the Norwegian scientist Nils Aall Barricelli who was studying artificial life at the institute for advanced study in Princeton, NJ [306]. A few years later in 1958 the Australian quantitative geneticist Alex Fraser published his seminal work *"Simulation of genetic systems by automatic digital computers"*. Fraser's efforts in the 1950s and 1960s had a profound impact on the development of computational models of evolutionary systems. Another key player at the time was the American scientist Lawrence J. Fogel who is known as the father of evolutionary programming [307].

Though many researchers picked-up on the idea, it wasn't until the early 1970s when John Holland et. al at the University of Michigan introduced genetic algorithms (GA) and Ingo Rechenberg and Hans-Paul Schwefel from the Technical University of Berlin introduced evolution strategies, that EAs became widely recognized. These areas developed separately for about 15 years and were joined by genetic programming in the 1980s (Stephen F. Smith (1980) [308], Nichael L. Cramer (1985) [309], D. Dickmanns (1987) [310]) and 1990s (John R. Koza [311]).

EAs are part of a wider class of biologically inspired algorithms (sometimes referred to as *soft computing*). Other members of this class include neural networks, fuzzy theory, Bacteriologic Algorithms, Harmony Search, Ant Colony Optimization and Particle Swarm Optimization.

7.3.2 Important remarks

Before we continue, the reader should be reminded that EAs - like other soft computing techniques (e.g., neural nets) - are extreme simplifications of their biological counterparts and results/conclusions obtained in the artificial setting can usually not be generalized to the biological setting. In addition it cannot be stressed too strongly that an EA is not a random search for a solution to a problem. EAs use stochastic processes, but the result is distinctly non-random (better than random) [312]. Another common misconception is that EAs don't require any structure in the search space. This is definitely not the case, especially if a recombination operator is involved. Finally, in the context of optimization EAs are often used with the argument that they are able to find the global optimum [313]. This is of course not true. An EA may increase the chance of finding a global optimum if adequate operators are defined (the importance of which many users underestimate) but it should never be blindly trusted to do so.

7.3.3 Types

Five major types of EAs can be identified [312]:

1. Genetic Algorithms
2. Evolutionary Programming
3. Evolution Strategies
4. Classifier Systems
5. Genetic Programming

Gray zones exist between the different classes but all share a common conceptual base of simulating the evolution of individual structures via processes of selection, recombination, mutation and reproduction. These processes are driven by the performance of the individual structures as defined by an environment (fitness function). The art of applying EAs is finding a good balance between exploration (global search) and exploitation (local search) when combining the different processes. In this thesis we are concerned with genetic algorithms: a population of individuals, represented by their genome, is evolved through the use of selection, recombination (crossover) and mutation operators for a fixed number of generations.

7.4 The Genetic Algorithm

7.4.1 The Canonical GA

The GA is probably the most well known EA and is used as an algorithm for global search, optimization being the most obvious application. The core algorithm, as introduced by Holland [314] is referred to as the Canonical Genetic Algorithm (CGA) and is presented in pseudo code below [312]:

```
//start with an initial time
t := 0;

//initialize a random population of individuals
initpopulation P(t);

//evaluate fitness of all initial individuals of populatio
evaluate P(t);

//test for termination criterion (time, fitness, etc.)
while not done do
   //increase the time counter
   t := t + 1;
   //select a sub-population for offspring production
   P' := selectparents P(t);
   //recombine the "genes" of selected parents
   recombine P'(t);
   //perturb the mated population stochastically
   mutate P'(t);
   //evaluate its new fitness
   evaluate P'(t);
   //select the survivors from actual fitness
   P := survive P,P'(t);
end
```

We adopt the notation from [315]. The population of the CGA consists of an n-tuple of binary strings b_i of length l, where the bits of each string are considered to be the *genes* of an individual chromosome. Each individual b_i represents a feasible solution in the search space with the quality of the solution determined by a fitness function f. Selection of individuals to reproduce is performed proportional to their fitness. The probability that individual b_i is selected from tuple $(b_1, b_2, ..., b_n)$ to be a member of the next generation is given by

$$P\{b_i \text{ is selected}\} = \frac{f(b_i)}{\sum_{j=1}^{n} f(b_j)} > 0 \qquad (7.1)$$

The population is initialized with random bit strings and individuals are modified by crossover and mutation operators. Mutation operates independently on each b_i by randomly flipping one or more bits. The event that the j-th bit of the i-th individual is flipped is stochastically independent and occurs with probability p_m. Crossover is applied to randomly paired individuals with probability p_c. Classically *single-point crossover* is used, a randomly chosen recombination point is chosen and crossover on the following two parents

$$\begin{array}{c} 111\,|\,11111 \\ 000\,|\,00000 \end{array}$$

produces the following two offspring

$$\begin{array}{c} 11100000 \\ 00011111 \end{array}$$

7.4.2 Extensions to the CGA

The CGA presented in the previous subsection is the GA in its simplest form. As such it is useful for studying the theoretical properties of GAs but for most applications it is extended in one or more of the following ways:

- non-bit string representation (e.g., integers, floating point numbers, character strings, graphs, etc.)
- adaptive parameters (e.g., varying mutation rates, crossover rates, population size, etc.)
- selection functions (e.g., tournament selection, stochastic universal sampling, etc.)
- speciation (*see* section 7.5)

In addition the GA may be combined with Lamarckian learning by performing a local optimization on every generated individual. This approach is referred to as a memetic algorithm [316].

7.4.3 Theoretical foundations

The fundamental theorem in GAs is Hollands *Schema Theorem* [314]. A schema is a bit string with one or more *don't care* values. For example, the schema *100*10* has one don't care value and represents two possible bitstrings *100110* and *100010*. The order of a schema s is defined as the number of non-don't care positions. E.g., in the previous example the order $o(s) = 5$. The defining length δ of a schema is defined as the distance between the first and the last fixed string positions. It defines the compactness of information in a schema. For example, for $s=$****001*110*, $\delta(s) = 10 - 4 = 6$.

Given these definitions the Schema Theorem can be stated as follows:

Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm.

Mathematically, this can be formulated as:

$$\xi(S,t+1) \geq \frac{\xi(S,t)}{\overline{F(t)}} \cdot \left[(1-p_c) \cdot \frac{\delta(S)}{l-1} - o(S) \cdot p_m\right]$$

where $\overline{F(t)}$ is the average fitness of the population and $\xi(S,t)$ is defined as the expectation of the number of bit strings matching schema S at time t. Based on the Schema Theorem Goldberg [307] proposes the Building Block Hypothesis [317]:

A genetic algorithm seeks near-optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks.

Goldberg states in [307];

Short, low-order, and highly fit schemata are sampled, recombined, and re-sampled to form strings of potentially higher fitness. In a way, by working with these particular schemata (the building blocks), we have reduced the complexity of our problem; instead of building high-performance strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings.

So, the schema theorem says that the GA will produce increasingly fit individuals where the better individuals match short, low-order schemata. However, the main question is, does the CGA converge to the globally best solution? Intuitively we can already see a problem if we consider a search space where the best individual is a long, high order schema (e.g., 01111011). Crossover and mutation will easily break the schema thus the GA may never find the best individual. Work by Rudolph in [315] confirms that this is the case, convergence of the CGA cannot be guaranteed. They prove that, to ensure convergence, an elitist GA must be used where the best individual found over time is manually preserved. However Rudolph's proofs only prove that the global solution can be found, they say nothing about the time needed to reach a solution.

The Schema theorem proves that the CGA will make progress when searching the parameter space. However, the theorem is plagued with a number of problems that limit its practical application:

- Proof of convergence
- Only applicable to the CGA (bit string representation)
- The Schema theorem is an inequality instead of an equality[3], it only provides a lower bound for the expected number of schema's. This makes it difficult to

[3]The theorem neglects the small probability that a string belonging to the schema s will be created from nothing by a mutation of a string that did not belong to s in the previous generation.

use schema theories to predict the future behavior of a GA even for a single generation ahead [318]

Though many extensions to the theorem have been developed, the schema theorem and its variants have been widely criticized with many researchers believing that "*schema theorems are nothing more than trivial tautologies of no use whatsoever*" [318]. This is arguably even more so for the Building Block Hypothesis, where Goldberg himself admits that "*While these claims seem perfectly reasonable, how do we know whether they hold true or not*" [307]. This point will be revisited in section 7.7.

7.4.4 Applications

Theoretical foundations aside, GAs have found widespread use in many domains. The main disadvantage of GAs are the high number of function evaluations needed for convergence. Therefore they are best suited to problems where there is little problem specific information that can be exploited and where traditional, usually gradient-based, algorithms perform poorly. Such problems are typically characterized by high noise, non-polynomial time complexity, many local minima (multi-modal), and dependencies between variables (epistasis). Within this domain GAs have performed very successful with many applications in transportation [319], electronics [320], vehicle design [321], scheduling [322], data fitting [323], and many others.

7.5 Parallel Genetic Algorithms

7.5.1 Introduction

Since GAs are population-based they easily lend themselves to parallelism. The total population can be divided into different sub-populations evolving in parallel, each scheduled on a different CPU. The motivation for dividing up the population need not be a purely computational one. For example, from a biological standpoint it makes sense to consider *speciation:* Genomes that differ considerably from the rest of the population are automatically split off into a separate sub-population and continue to evolve independently. Thus forming a new species. In this way the parameter space is searched more efficiently. The idea of speciation, like many other concepts and operators, was pioneered by Holland in the early seventies [314].

The terms Parallel Genetic Algorithms (PGA) or Distributed Genetic Algorithms (DGA) [324] usually refer to the case whenever the population is divided up in some way, for whatever reason. Strictly speaking the terms refer to the actual implementation of the GA on (massively) parallel hardware or on a grid (e.g., [325]). Unfortunately though, the terminology for the different models varies between authors and can be very confusing [326].

In this chapter we are only interested in PGA on the model level (i.e., different speciation models), parallelism for computational reasons, including scheduling, will not be considered.

7.5.2 Island model

The island model [327] is probably the most well known PGA. Different sub-populations exist (initialized differently) and sporadic migration can occur between islands allowing for the exchange of genetic material between species and inter-species competition for resources.

This model is also known as the *migration model* [328] or *stepping stone model* [326], depending on the migration constraints. As stated above, the population is divided into a number of independent sub-populations, so-called *demes*, with inter-deme migration. Selection and recombination are restricted per deme, such that each sub-population may evolve towards different locally optimal regions of the search space (called *niches* in the terminology of Goldberg [307]). Depending on the size and number of demes the model can be *coarse grained* or *fine grained* [326]. The migration model introduces five new parameters: the migration topology, the migration frequency, the number of individuals to migrate, a strategy to select the emigrants, and a replacement strategy to incorporate the immigrants. The island model is illustrated in figure 7.3 for two topologies.

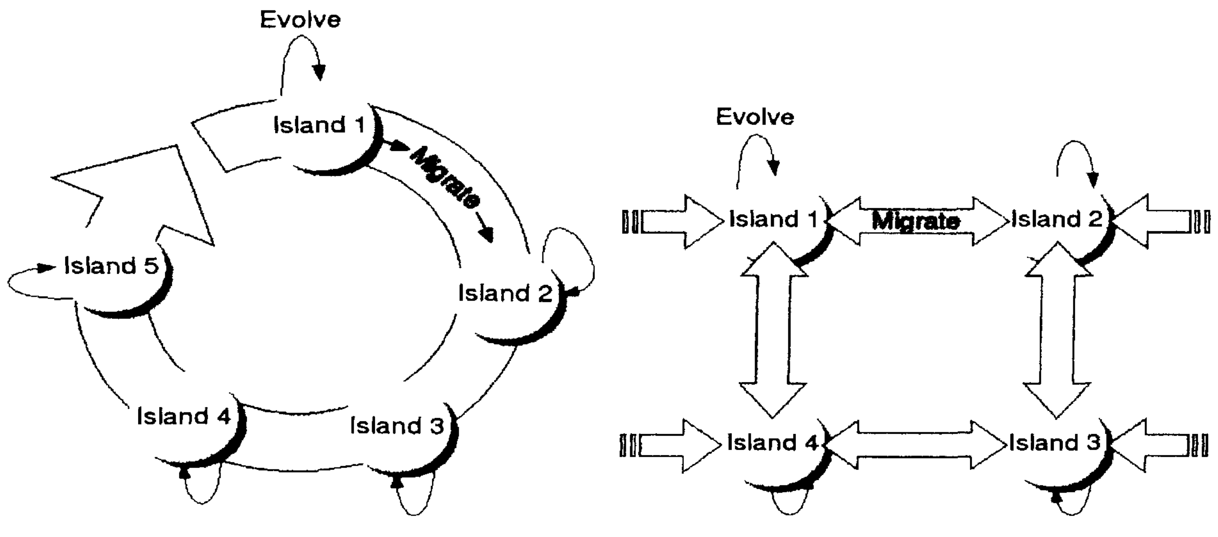


Figure 7.3: Ring and grid migration topologies in the Island Model

A famous real world example of this are Darwin's finches (also known as the Galapagos Finches). These are 13 or 14 different but closely related species of finches Charles Darwin collected on the Galapagos Islands during the Voyage of the Beagle. Darwin later established that each species was uniquely related to individual islands. The geographical isolation was such that each species could adapt to the environment on its specific island, from a common ancestor, while still being able to migrate to a different island. The following quote from chapter 17 of Darwin's *The Voyage of the Beagle* illustrates this:

> *The remaining land-birds form a most singular group of finches, related to each other in the structure of their beaks, short tails, form of body and plumage: there are thirteen species, which Mr. Gould has divided into four subgroups. All these species are peculiar to this archipelago; and so is the whole group, with the exception of one species of the sub-group Cactornis, lately brought from Bow Island, in the Low Archipelago. [...] The most curious fact is the perfect gradation in the size of the beaks in the different species of Geospiza, from one as large as that of a hawfinch to that of a chaffinch, and (if Mr. Gould is right in including his sub-group, Certhidea, in the main group) even to that of a warbler. The largest beak in the genus Geospiza is shown in Fig. 1, and the smallest in Fig. 3; but instead of there being only one intermediate species, with a beak of the size shown in Fig. 2, there are no less than six species with insensibly graduated beaks. The beak of the sub-group Certhidea, is shown in Fig. 4. The beak of Cactornis is somewhat like that of a starling, and that of the fourth subgroup, Camarhynchus, is slightly parrot-shaped. Seeing this gradation and diversity of structure in one small, intimately related group of birds, one might really fancy that from an original paucity of birds in this archipelago, one species had been taken and modified for different ends. In a like manner it might be fancied that a bird originally a buzzard, had been induced here to undertake the office of the carrion-feeding Polybori of the American continent.*

"Mr. Gould" from the quote refers to John Gould, a famous English ornithologist.

7.5.3 Cellular model

Another model is the *cellular model* [329] (also known as the *diffusion* model [328] or *massively parallel GA* [326]). Instead of parallelism on the population level, the diffusion model concentrates on interactions of individuals within a single population. In this case parallelism is performed on the level of individuals. Communication (selection, recombination) of individuals is restricted to a local neighborhood structure. This type of separation is referred to as *isolation by distance* [330]. This way, advantageous genetic information may arise at different points in the topological interaction structure and spread slowly over the population. In this case the neighborhood size and the interaction structure play an important role for maintaining diversity [328]. While there are no explicit islands, there is the possibility of similar effects.

The cellular model is illustrated in figure 7.4 for a neighborhood distance of one.

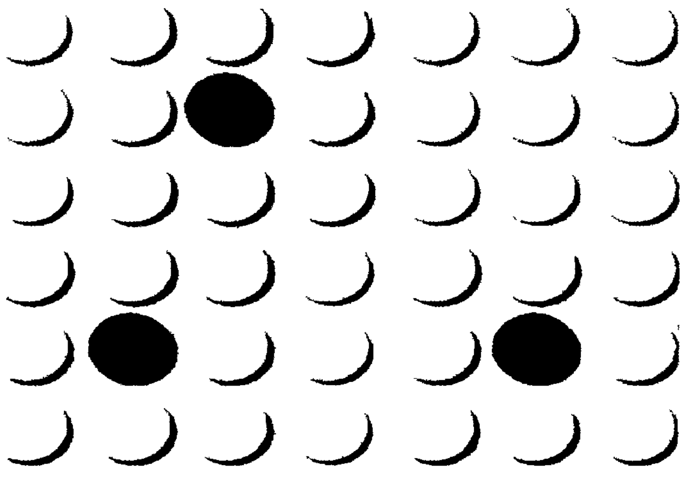


Figure 7.4: Cellular speciation model

7.5.4 Fitness sharing

A third model, the one originally proposed by Holland [314] and applied to the 2-arm bandit problem, revolves around the *sharing* concept. It is inspired from the observation of positive assortive mating in nature (like mates like). The model is commonly known as *fitness sharing.* A sharing function $s(d)$ is defined to determine the neighborhood and degree of similarity between each individual in the population (e.g.: if individuals are bit strings, $s(d)$ can be defined as $s(d) : \mathbb{N} \mapsto [0, 1]$, with d proportional to the Hamming distance). Each individual that belongs to the same species (as determined by $s(d)$) then receives the same fitness value. See for example the overview in [331].

7.5.5 Others

The three model types listed here are the major categories, though many variations exist such as: inbreeding with intermittent crossbreeding, overlapping demes, dynamic demes, segregative GA, crowding, pre-selection, co-evolutionary algorithms, hierarchical GA, Cohort GA, the community model and the plant pollination model. Additionally, many hybrid schemes are possible where different aspects of each model are combined. See for example [332].

7.5.6 Applications

The different speciation models have also found widespread use. For example, [333] uses speciated evolution for inference of Bayesian networks. Another example is NEAT [334], a platform that uses fitness sharing to evolve neural networks. Other uses include scheduling [335, 336], surrogate driven optimization [337] (using a hierarchy of island models), and vehicle concept selection in aerospace [338]. An extensive treatment of all the applications of EA is out of scope for this chapter, excellent references can be found in [326,339,340].

7.6 Heterogeneous Evolution of Surrogate Models

This Section discusses how different surrogate models may be evolved cooperatively in order perform model type selection. For this it is important for the reader to revisit the general global surrogate modeling control flow described in chapter 3 since it forms the basis for the evolutionary algorithm described below.

7.6.1 Motivation

While the mathematical formulation of global surrogate modeling presented in section 3.2 is clear cut, its practical implementation raises a number of obvious questions and design choices. These are discussed in detail in chapter 3. For this chapter however, we are most interested in two particular subproblems: the model type selection problem (section 3.7.1) and the model complexity selection problem (section 3.7.2).

7.6.1.1 Classic approach

If multiple model types are considered, the classic approach is to simply to try out different types and select the best one according to one or more accuracy criteria. There is ample literature available that benchmarks model types in this way: [2, 3, 10–20]. But claims that a particular model type is superior to others should always be met with some skepticism.

In order for the different benchmarking studies to be truly useful for a domain expert, the results of such studies must be collected and compiled into a general set of rules, recipe, or flowchart. To ease the discussion, let us denote such a compilation into a learning algorithm by L. L is then essentially a classifier that can predict which model type $t \in T$ to use based on data D and application requirements Γ:

$$L(D,\Gamma) = t \tag{7.2}$$

When executed L should then be able to give a specific recommendation as to which model type to use for a given problem. This recommendation should be more specific than the general rules of thumb that are available now. Experience shows this to be exactly what an application engineer wants. However, constructing such a learner L for any but the most restricted class of problems is a daunting undertaking for obvious practical reasons. Firstly, deciding which problem/application features to train the classifier on is far from trivial. Also even if this is done, the number of features can be expected to be high thus gathering the necessary data (by manually solving equation 3.2) to train L accurately will be very computationally expensive.

Secondly, as mentioned above, the success of a model type largely depends on the expertise of the user, the quality of the data, and even the quality of the software implementation of the technique. Neural networks are a good example in this respect. In the right hands they are able to perform very well on many problems. However, if poor choices are made with regard to training function, topology selection, generalization control, training parameters, software library, etc. they may seem to perform poorly. How to take into account this information in L?

Thirdly, a more fundamental problem with this approach is that data must be available in order for the reasoner to work. However, if only a simulation code is available (as is often the case) data must be collected, and the optimal data collection strategy that minimizes the number of points depends on the model type. Also, the optimal model type will change depending on how much data is available and how it is distributed [341]. One could argue to instead train L only on the data characteristics which are known in advance (e.g., dimensionality, noise level, etc.). The question is then again, which characteristics are most important? Furthermore, in many cases not much is known about the true behavior of the response thus there will typically not be enough information to train L accurately.

This brings us to the final point. A main reason for turning towards global surrogate modeling methods is that little is known about the behavior of the response [44]. The goal is to get insight into that behavior in a computationally cheap way by applying surrogate methods. Another reason why information about the data may be scarce is that the source of the data is confidential or proprietary and very little information is disclosed. In these situations using or training L becomes very difficult.

Finally, we must stress that we do ***not*** say that this problem is too difficult and not worth trying to solve. Indeed many such problems exist and are currently being tackled, particularly in medicine. Instead we argue that users of global surrogate modeling methods can benefit from a more dynamic approach that is flexible, can be easily applied to a wide range of different problems, can easily incorporate new fitting techniques and process knowledge, and naturally integrates with an adaptive data collection procedure. We shall revisit this point in sections 7.6.1.2 and 7.7.

Assuming the model type selection problem has been solved, there remains the model parameter selection problem. This particular problem has been discussed in detail in section 3.7.2 and is not the focus of this chapter

7.6.1.2 Proposed solution

While we are primarily interested in the first problem, the approach described in this chapter naturally incorporates problem 2 as well. In both cases there is little theory that can be used as a guide. It is in this setting that the evolutionary approach can be expected to do well. We describe the application of a single GA with speciation to both problems: the selection of the surrogate type and the optimization of the surrogate model parameters (hyperparameter optimization). In addition, we do not assume all data is available at once but must be sampled incrementally since it is expensive (active learning).

The idea is to maintain a heterogeneous population of surrogate model types and let them evolve cooperatively and dynamically with the changing data distribution. The details will be presented below.

7.6.2 Related Work

The evolutionary generation of regression models for given input-output data has been widely studied in the genetic programming community [342–344]. Given a set of mathematical primitives (+, *sin*, *exp*, /, *x*, *y*, etc.) the space of symbolic expression trees is searched to find the best function approximation. The application of GAs to the optimization of model parameters of a single model type (homogeneous evolution) has also been common: [345–349] and the extensive work by Yao et. al. [350, 351]. Integration with adaptive sampling has also been discussed [22]. However, these efforts do not tackle the model *type* selection problem, they restrict themselves to a particular method (e.g., SVMs or neural networks). As [262] state "*Little is known about which types of model accord best with particular features of a landscape and, in any case, very little may be known to guide this choice.*". Likewise, [67] note: "*...it is important to stress that there are always situations when one model type cannot be applied or suffers from inadequacies and can be well complemented or replaced by another one*". Thus an algorithm to help solve this problem in a dynamic, automated way is very useful [352]. This is also noticed by [261] who compare different surrogate models for approximating each objective during optimization. They note that in theory their approach allows the use of a different model type for each objective. However, such an approach will still require an a priori model type selection and does not allow for dynamic switching of the model type or the generation of hybrids.

There has also been much research on the use of surrogate models in evolutionary optimization of expensive simulators (to approximate the fitness function). References include [353–356], the work by Ong et al. [357], and more recently by Lim et. al. [18]. In general the theory is referred to as Surrogate Based Optimization or Metamodel Assisted Optimization. A good overview reference is given by [63] and [12]. For example, [18] compare the utility of different local surrogate modeling techniques (quadratic polynomials, GP, RBF, ...) including the use of (fixed) ensembles, for optimization of computationally expensive simulation codes. Local surrogates are used together with a trust region framework to quickly and robustly identify the optimum. As noted in the introduction, the contrast with this work is that references such as [18] are interested in the optimum and not the surrogate itself (they make only a "*mild assumption on the accuracy of the metamodeling technique*"). In addition the model parameters are taken as fixed and there is no integration with active learning. In contrast we place very strong emphasis on the surrogate model accuracy, the automatic setting of the hyperparameters, and the efficient sampling of the *complete* design space.

The work of Sanchez et. al [358] and Goel et. al [87] is more useful in our context since they provide new algorithms for generating an optimal set of ensemble members for a fixed set of data points (no sampling). Unfortunately, though, the parameters of the models involved must still be chosen manually. Nevertheless, their approaches are interesting, and can be used to further enhance the approach presented here. For example, instead of returning the single final best model, an optimal ensemble member selection algorithm can be used to return a potentially much better model based on the final population or Pareto front.

The work in [359] from machine learning is also related. The authors describe an interesting classification algorithm *COMB* that combines online an ensemble of active learners so as to expedite the learning progress in pool-based active learning. In their terminology an active learner is a combination of a model type and a sampling algorithm. A weighted ensemble of active learners is maintained and each learner is allowed to express interest in a pool of unlabeled training points. Depending on the interests of the active learners, an unlabeled point is selected, labeled by the teacher, and based on the added value of that point the different active learners are punished or rewarded. Internally the active learners are SVM models whose parameters are chosen manually. In principle, with a number of approximations one could adapt the algorithm to the regression case. If one then also included hyperparameter optimization, the result would be very similar to the SUMO Toolbox (cfr. chapter 4) configured with one or more of the *Error*, LRM, or EGO [89] sample selection algorithms, but without the ability to combine different criteria. However, a problem would be that *COMB* assumes a pool of unlabeled training data is given. However, when modeling a simulation code in regression no such pool is available. Some external algorithm would still be needed to generate it in order for *COMB* to work. *COMB* does also naturally allow for different model types but in a more static way than the algorithm in Section 7.6.3: there is no hyperparameter optimization, the number of each active learning type remains fixed (though the weights can change) leading to a potentially high computational cost, and hybrid models are not considered. The extension to the multi-objective case is also non-trivial. Of course *COMB* could be extended to incorporate such features, but the result would be very similar to the work presented here. Nevertheless, the specific scoring functions, probability weightings, and ensemble weight updates, seem very useful and could be implemented in the SUMO Toolbox to complement the approach presented here.

Finally, the work by Escalante et. al. [360] is most similar to the topic of this chapter. [360] consider the problem of finding the optimal classifier and associated hyperparameters for a given classification problem (active learning is not considered). A solution is encoded as a vector and Particle Swarm Optimization (PSO) is used to search for good classifiers. Good results are shown on various benchmarks. Unlike the GA approach, however, it is less straightforward to deal with multiple sub-populations, giving models room to mature independently before entering competition. The use of operators tuned to specific models is also difficult (to increase the search efficiency). In effect, the PSO approach takes a top-down view, using a high level encoding in a high dimensional space, a typical particle has 25 dimensions [361]. In contrast the GA approach is bottom up, the model specific operators result in a much smaller search space, different for each method (e.g., 1 for the spline models and 2 for the SVM models). This leads to a more efficient search requiring less fitness evaluations and facilitates the incorporation of prior knowledge. In addition, by using PSO there is no natural way of enabling hybrid solutions (ensembles) without extending the encoding and further increasing the search space. In contrast, the hybrid solutions arise very naturally in the GA framework and do not impact the search space of the other model types. The same is true of the extension to the multi-objective case, a very natural step in the GA case.

In sum, in by far the majority of the related work considered by the authors, speciation was always constrained to one particular model type, for example neural networks in [334]. The model type selection problem was still left as an a-priori choice for the user. Or, if multiple model types are used, the hyperparameters are typically kept fixed and there is no tie-in with the active learning process.

7.6.3 Algorithm

We now present the concrete GA for heterogeneous evolution as it is embedded (as a plugin) in the SUMO Toolbox. The speciation model used is the island model since we found it the most natural way of evolving multiple model types while still allowing for hybrid solutions. The algorithm is based on the Matlab GADS toolbox and works as follows (see figure 7.5 and reference [273] for more details): After the initial DOE has been calculated (cfr. the control flow in Section 4.5.1), an initial sub-population M_i is created for each model type $t \in T$ $(i = 1,..,h)$. The exact creation algorithm is different for each model type so that model specific knowledge can be exploited. Subsequently, each deme is allowed to evolve according to an elitist GA. Parents are selected according to a selection algorithm (e.g., tournament selection) and offspring undergo either crossover (with probability p_c) or mutation (with probability $1 - p_c$). The models M_i are implemented as Matlab objects (with full polymorphism) thus each model type can choose its own representation and mutation/crossover implementations (this implements the minimization over $\theta \in \Theta$ of equation 3.2). While mutation is straightforward, the crossover operator is more involved (see Section 7.6.5 below).

The fitness function calculates the quality of the model fit, according to criteria ξ. The current deme population is then replaced with its offspring together with el elite individuals. Once every deme has gone through a generation, migration between

```
$X_0 = initialExperimentalDesign()$;
$X = X_0$;
$f|_X = evaluateSamples(X)$;
$T = \{t_1, ..., t_h\}$;
$M_i$ = createInitialModels($t_i, popsize_i$); $i = 1, .., h$
$M = \bigcup_{i=1}^{h} M_i$;

while($\xi$ not reached) do
  $scores = \{\}$;  $gen = 1$;
  while($termination\_criteria$ not reached) do
    foreach $M_i \subseteq M$ do
      $scores_i$ = fitness($M_i, X, f|_X, \xi$);
      $elite$ = sort([$scores_i; M_i$])$|_{1:el}$;
      $parents$ = select($scores_i, M_i$);
      $parents_{xo}$ = selectXOParents($parents, p_c$);
      $offspring_{xo}$ = crossover($parents_{xo}, ES_{diff}, ES_{max}$);
      $parents_{mut} = parents \backslash parents_{xo}$;
      $offspring_{mut}$ = mutate($parents_{mut}$);
      $M_i = elite \bigcup offspring_{mut} \bigcup offspring_{xo}$;
      $scores = scores_i \cup scores$;
    end
    if($mod(gen, m_i) = 0$)
      $M$ = migrate($M, scores, m_f, m_d$)
    end
    $M$ = extinctionPrevention($M, T_{min}$);
    $gen = gen + 1$;
  end
  $X_{new}$ = selectSamples($X, f|_X, M$);
  $f|_{X_{new}}$ = evaluateSamples($X_{new}$);
  $[X, f|_X] = merge(X, f|_X, X_{new}, f|_{X_{new}})$;
end

return bestModel($M$);
```

Figure 7.5: Algorithm for global surrogate modeling with heterogeneous evolution and active learning

individuals is allowed to occur at migration interval m_i, with migration fraction m_f and migration direction m_d (a ring topology is used). The migration strategy is as follows: the $l = (|M_i| \cdot m_f)$ fittest individuals of M_i replace the l worst individuals in the next deme (defined by m_d). As in [362], migrants are duplicated, not removed from the source population. Note that in this contribution we are primarily concerned with inter-model speciation (speciation as in different model types). Intra-model speciation (e.g., through the use of fitness sharing within one model type) is something which was not done but could easily be incorporated.

Once the GA has terminated, control passes back to the main global surrogate modeling algorithm of the SUMO Toolbox. At that point M contains the best set of models that can be constructed for the given data. If the accuracy of the models is sufficient the main loop terminates. If not, a new set of maximally informative sample points is selected based on several criteria (quality of the models, non-linearity of the response, etc.) and scheduled for evaluation. Once new simulations become available the GA is resumed.

Note that sample evaluation and model construction/hyperparameter optimization run in parallel. For clarity, algorithm 7.5 shows them running sequentially but this is not what happens in practice. In reality both are interleaved to allow an optimal use of computational resources.

7.6.4 Extinction prevention

Initial versions of this algorithm exposed a major shortcoming, specifically due to the fact that models are being evolved. Since not all data is available at once but trickles in, $|X_j| - |X_{j-1}|$ samples at a time, models that need a reasonable-to-large number of samples to work well will be at a huge disadvantage initially. Since they perform badly at first, they may get overwhelmed by other models who are less sensitive to this problem. In the extreme case where they are driven extinct, they will never have had a fair chance to compete when sufficient data *does* become available. They may even have been the superior choice had they still been around[4]. Therefore an Extinction Prevention (EP) algorithm was introduced that ensures a model type can never disappear completely.

EP works by monitoring the population and each generation recording the number of individuals of each model type. If this number falls below a certain threshold T_{min} for a certain model type, the EP algorithm steps in and ensures the model type has its numbers replenished up to the threshold. This is done by re-inserting the last models that disappeared for that type (making copies if necessary). The re-inserted models replace the worst individuals of the other model types (who do have sufficient numbers) evenly.

Strictly speaking, EP goes completely against the survival of the fittest principle in evolutionary algorithms. By using it we are manually working against selection, preserving model types which give poor results at that point in time. However, in this setting it seems a fair measure to take (we do not want to risk losing a model type completely) and improves results in most cases (*see* Section 7.11). At the same time it is straightforward to implement and understand, needing no special control parameters. All it has to ensure is that a species is never driven extinct.

[4]As an example, this observation was often made when using rational models on electro-magnetic data.

7.6.5 Heterogeneous recombination

The attentive reader will have noticed that one major problem remains with the implementation as discussed so far. The problem lies in the genetic operators, more specifically in the crossover operator. Migration between demes means that model types will mix. This means that a set of parents selected for reproduction may contain more than one model type. The question then arises: how to perform recombination between two models of completely different types. For example, how to meaningfully cross an Artificial Neural Network with a rational function? The solution we propose here is to use ensembles (behavioral recombination). If two models of different types are selected to recombine, an ensemble is created with the models as ensemble members. Thus, as soon as migration occurs, model types start mixing, and ensemble models arise as a result. These are treated as a distinct model type just as the other model types.

However, the danger with this approach is that the population may quickly be overwhelmed by large ensembles containing duplicates of the best models (as was noticed during initial tests). To counter this phenomenon we apply the similarity idea from Holland's sharing concept [314]. Individual models will try to mate only with individuals of the same type. Only in the case where selection has made this impossible shall different model types combine to form an ensemble. In addition we enforce a maximum ensemble size ES_{max} and require that ensemble members must differ ES_{diff} percent in their response (their 'behavior'). This is calculated by evaluating the models on a dense grid.

This leaves us with only three cases left to explain:

1. *ensemble - ensemble* recombination: a single-point crossover is made between the ensemble member lists of each model (note that the type of the ensemble members is irrelevant)
2. *ensemble - model* recombination: the model replaces a randomly selected ensemble member with probability p_{swap} or gets absorbed into the ensemble with probability $1 - p_{swap}$ (respecting ES_{max} and ES_{diff}).
3. *ensemble* mutation: one ensemble member is randomly deleted

Besides enabling hybrid solutions, using ensembles has the additional benefit of allowing a model to lie 'dormant' in an ensemble with the possibility of re-emerging later (e.g., if after mutation only one ensemble member remains). Note that, in contrast to [18] for example, the type of the ensemble members is not fixed in any way but varies dynamically.

We have not yet mentioned what type of ensemble will be used. There are several methods for combining the outputs of models, such as average, weighted average, Dempster-Shafer methods, using rank-based information, supra-Bayesian approach, stacked generalization, etc [202]. To keep the implementation straightforward and the complexity (number of parameters) low we have opted for a simple average ensemble.

Of course different, more powerful combination methods could be used instead and they will only improve results. The exact method used is of lesser importance since it does not change the methodology. The advantage of a simple average ensemble is that it works in all cases: It makes no assumption on the model types involved, nor does it mandate any changes to the models or training algorithms (for example, like negative correlation learning) since this is not always possible (e.g., when using proprietary, application specific, modeling code).

7.7 Critique

The algorithm presented so far has a number of strengths and weaknesses. The obvious advantage is the ability to perform automatic selection of the model type and complexity for a given data source (no need to do multiple parallel runs or train a complex classifier). In addition the algorithm is generic in that it is independent of the data origin (application), model type, and data collection strategy. New approximation methods can easily be incorporated without changing the algorithm. Problem specific knowledge and model type specific optimizations based on expert knowledge can also be incorporated if needed (i.e., by customizing the genetic operators). Furthermore, the algorithm naturally integrates with the data collection strategy, allowing the best model type to change dynamically and naturally allows for hybrid solutions. Finally, it naturally extends to the multi-objective case and can be easily parallelized to allow for faster computations (though the computational cost is still outweighed by the simulation cost).

The main disadvantage is due to the fact that the approach is based on evolutionary algorithms: full determinism cannot be guaranteed. This raises the obvious question of how stable the convergence is over multiple runs. The same can be said of standard approaches towards hyperparameter optimization (which typically include randomization) or for any algorithm involving a GA for that matter. Formulating theoretical foundations in order to come to convergence guarantees for GAs is a difficult undertaking and has been the topic of intense research ever since their inception in the late 80s. Characterizing the performance of genetic algorithms is complex and depends on the application domain as well as the implementation parameters [363]. Most theoretic work has been done on schema theorems for the Canonical Genetic Algorithm (CGA), which try to prove convergence in a simplified framework using a binary representation. However prediction of the future behavior of a GA turns out to be very difficult and much controversy remains over the usefulness of these theorems [307, 318]. Theoretical work on other classes of GAs or using specific operators has also been done [363–367] but is unfortunately of little practical use here. For example, the work in [368] requires the calculation of fitness ratios, but this is impractical (and computationally expensive) to do in this situation and the results will vary with the application.

Thus, for the purposes of this chapter a full mathematical treatment of algorithm

7.5 and its convergence is out of scope. Due to the island model, sampling procedure, and heterogeneous representation/operators used, such a treatment will be far from trivial to construct and distract from the main theme of the chapter. In addition its practical usefulness would remain questionable due to the many assumptions that will be required. However, gaining a deeper theoretical insight into the robustness of the algorithm is still very important. A sensitivity study, as described in [369], of the main GA parameters involved will shed more light on this issue.

Theoretical remarks aside, we found that in practice the approach works quite well. If reasonable population sizes are used together with migration and the extinction prevention algorithm described in Section 7.6.4, the results of the algorithm are quite robust and give useful results and insights into the modeling problem. Besides the results given in this chapter, good results have also been reported on various real world problems from aerodynamics [370], electronics [267], hydrology [371], and chemistry [273].

In sum, this approach is useful if: little information is known about the expected structure of the response, if it is unclear which model type is most suited to the problem, data is expensive and must be collected iteratively, and hybrid solutions are useful. In other cases, for example, it is clear from a priori knowledge which model type will be the most suitable (e.g., based on existing rules of thumb for a well defined, restricted problem), this approach should not be applied, save as a comparison.

7.8 Test Problems

We now consider five test problems to which we apply the heterogeneous GA (from now on abbreviated by HGA). The objective is to validate if the best model type can indeed be determined automatically, and in a way that is cheaper and better than the simple brute force method: doing multiple, single model type runs in parallel. The problems include 2 predefined mathematical functions, and real-world problems from electronics and aerodynamics.

The dimensionality of the examples ranges from 2 to 13. This is no inherent limit but simply depends on the model types used. For example if only SVM-type models are used the number of dimensions can be arbitrarily high, while for smoothing spline models the dimensionality should be kept low. It all depends on which model types make up the population.

We also hope to see evidence of a 'battle' between model types. While initially one species may have the upper hand, as more data becomes available (dynamically changing hyperparameter optimization landscape) a different species may become dominant. This should result in clearly noticeable population dynamics, a kind of oscillatory stage before convergence. We briefly discuss each of the test problems in turn.

7.8.1 Ackley Function (AF)

The first test problem is Ackley's Path, a well known benchmark problem from optimization, a plot is shown in figure 7.6. Its mathematical definition for d dimensions is:

$$\begin{aligned} F(\vec{x}) &= -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{d} \cdot \sum_{i=1}^{d} x_i^2}\right) \\ &\quad - \exp\left(\frac{1}{d} \cdot \sum_{i=1}^{d} \cos(2\pi \cdot x_i)\right) + 20 + e \end{aligned} \tag{7.3}$$

with $x_i \in [-2,2]$ for $i = 1,\ldots,d$. For easy visualization we take $d = 2$. For this function a validation set and a test set of 5000 random points each is available. Although this is a function from optimization we are not interested in optimizing it, rather in reproducing it using a regression method with minimal data.

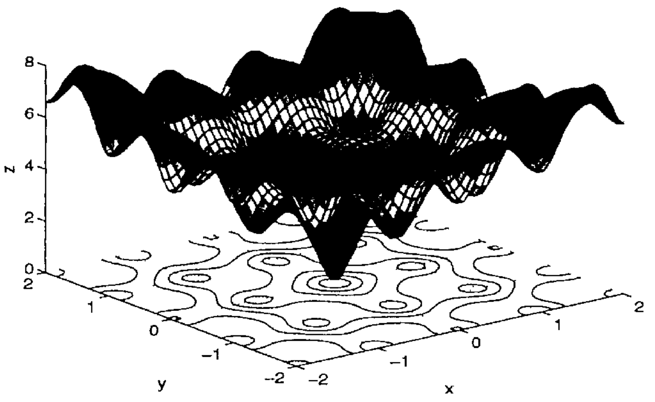


Figure 7.6: The Ackley Function

7.8.2 Kotanchek Function (KF)

The second predefined function is the Kotanchek function [372], depicted in figure 7.7. Its mathematical definition is given as:

$$F(x_1, x_2, u_1, u_2, u_3) = \frac{e^{-x_2^2}}{1.2 + x_1^2} + \varepsilon \tag{7.4}$$

with $x_1 \in [-2.5, 1.5]$, $x_2 \in [-1.0, 3.0]$, and with ε uniform random simulated numeric noise with mean 0 and variance 10^{-4}. As you can see only the first two variables are relevant. For this function a validation set and a test set of 5000 scattered points each is available.

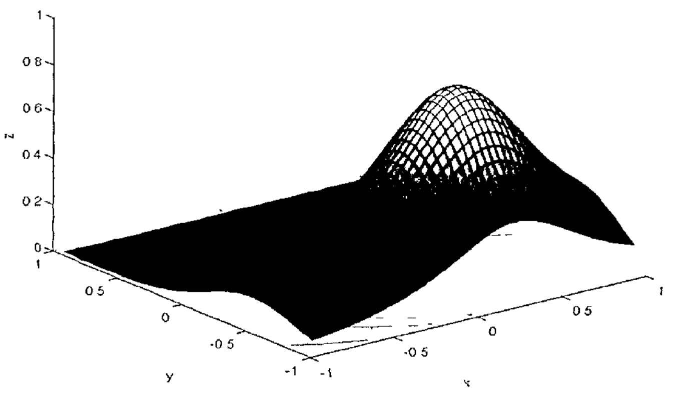


Figure 7.7: The Kotanchek function showing the two relevant variables

7.8.3 EM Example (EE)

The fourth example is a 3D Electro-Magnetic (EM) simulator problem [373]. Two perfectly conducting round posts, centered in the E-plane of a rectangular waveguide, are modeled, as shown in Figure 7.8. The 3 inputs to the simulation code are: the signal frequency f, the diameter of the posts d, and the distance between the two posts w. The outputs are the complex reflection and transmission coefficients S_{11} and S_{21}. The simulation model was constructed for a standard WR90 rectangular waveguide with $f \in [7 \text{ GHz}, 13 \text{ GHz}]$, $d \in [1 \text{ mm}, 5 \text{ mm}]$ and $w \in [4 \text{ mm}, 18 \text{ mm}]$. In addition, a 25^3 data set is available for testing purposes.

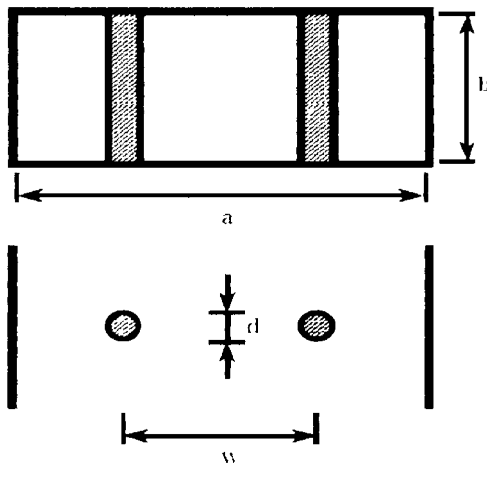
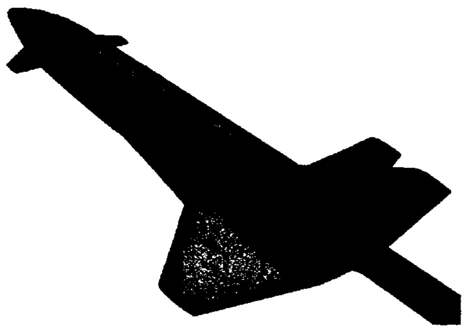
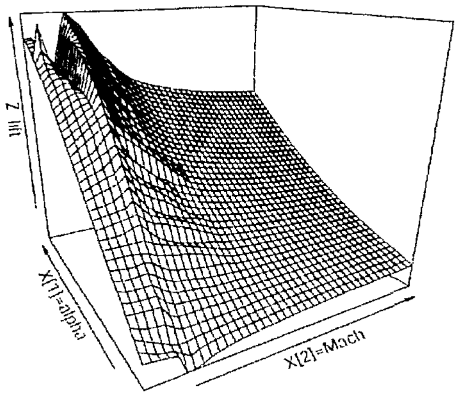


Figure 7.8: Cross sectional view and top view of the inductive posts [373]



(a) LGBB geometry [374]



(b) Lift plotted as a function of speed and angle of attack with side-slip angle fixed to zero. [375].

Figure 7.9: LGBB Example

7.8.4 LGBB Example (LE)

NASA's Langley Research Center is developing a small launch vehicle [374,376] that can be used for rapid deployment of small payloads to low earth orbit at significantly lower launch costs, improved reliability and maintainability. The vehicle is a three-stage system with a reusable first stage and expendable upper stages. The reusable first stage booster, which glides back to launch site after staging around Mach 3 is named the Langley Glide-Back Booster (LGBB). In particular, NASA is interested in the aerodynamic characteristics of the LGBB from subsonic to supersonic speeds when the vehicle reenters the atmosphere during its gliding phase.

More concretely, the goal is to gain insight about the response in lift, drag, pitch, side-force, yaw, and roll of the LGBB as a function of three inputs: Mach number, angle of attack, and side slip angle. For each of these input configurations the Cart3D flow solver is used to solve the inviscid Euler equations over an unstructured mesh of 1.4 million cells. Each run of the Euler solver takes on the order of 5-20 hours on a high end workstation [374]. The geometry of the LGBB used in the experiments is shown in Figure 7.9a.

Figure 7.9b shows the lift response plotted as a function of speed (Mach) and angle of attack (alpha) with the side-slip angle (beta) fixed at zero. The ridge at Mach 1 separates subsonic from supersonic cases. From the figure it can be seen there is a marked phase transition between flows at subsonic and supersonic speeds. This transition is distinctly non-linear and may even be non-differentiable or non-continuous [375]. Given the computational cost of the CFD solvers, the LGBB example is an ideal application for metamodeling techniques. Unfortunately access to the original simulation code is restricted. Instead a data set of 780 points chosen adaptively according to the method described in [375] was used.

7.8.5 Boston Housing Example (BH)

The Boston Housing dataset contains census information for 506 housing tracts in the Boston area and is a classic dataset used in statistical analysis. It was collected by Harrison et al and described in [377]. In the case of regression the objective is to predict the Median value of owner-occupied homes (in \$1000's) from 13 input variables (e.g., per capita crime rate by town, nitric oxide concentration, pupil-teacher ratio by town, etc.).

7.9 Model types

For the tests the following model types are used: Artificial Neural Networks (ANN), rational functions, RBF models, Kriging models, LS-SVMs, and for the AF example: also smoothing splines. For the EM example only the model types that support complex valued outputs directly (rational functions, RBF, Kriging) were included. Each type has its own representation and genetic operator implementation (thanks to the polymorphism as a result of the object oriented design). As stated in subsection 7.6.5 the result of a heterogeneous recombination will be an averaged ensemble. So in total up to seven model types will be competing to approximate the data. Remember that all model parameters are chosen automatically as part of the GA. No user input is required, the models and data points are generated automatically.

The ANN models are based on the Matlab Neural Network Toolbox and are trained with Levenberg Marquard backpropagation with Bayesian regularization [271, 272] (300 epochs). The topology and initial weights are determined by the GA. When run alone (without the HGA) this results in high quality models with a much faster run time than training the weights by evolution as well. Nevertheless, the high level Matlab code and complex training function do make the ANNs much slower than any of the other model types.

The LS-SVM models are based on the implementation from [269], the kernel type is fixed to RBF, leaving c and σ to be chosen by the GA. The Kriging model implementation is based on [270] (except for the EM example) and the correlation parameters are set by the GA (the regression function is set to linear and the correlation function to Gaussian). The RBF models (and the Kriging models for the EM example, since the data is complex valued) are based on a custom implementation where the regression function, correlation function, and correlation parameters are all evolved. The rational functions are also based on a custom implementation, the free parameters being the orders of the two polynomials, the weights of each parameter, and which parameters belong in the denominator. The spline models are based on the Matlab Splines Toolbox and only have one free parameter: the smoothness.

Remember that the specific model types chosen for the different tests is less important. This can be freely chosen by the user. What is important is rather how these different model types are used together in a single algorithm. Thus a full explanation of the virtues of each model types, as well as the representation and genetic operators used is out of scope for this chapter and would consume too much space. Details can be found in [273] or in the implementation that is available as part of the SUMO Toolbox.

7.10 Experimental setup

The following subsections describe the configuration settings used (and their motivation) for performing the experiments.

7.10.1 Sample selection settings

For the LE and BH examples only a fixed, small size, data set is available. Thus, selecting samples adaptively makes little sense. So for these examples the adaptive sampling loop was switched off. For the other examples the settings were as follows: an initial optimized Latin hypercube design, using the method from [242], of size 50 is used augmented with the corner points. Modeling is allowed to commence once at least 90% of the initial samples are available. Each iteration a maximum of 50 new samples are selected using the Local Linear (LOLA) adaptive sampling algorithm [239]. LOLA identifies new sample locations by making a tradeoff between eploration (covering the design space evenly) and exploitation (concentrating on regions where true response is nonlinear). LOLA's strengths are that it scales well with the number of dimensions, makes no assumptions about the underlying problem or surrogate model type, and works in both the $\mathbb{R}$ and $\mathbb{C}$ domains. LOLA is able to automatically identify non-linear regions in the domain and sample these more densely compared to more linear, 'flatter' regions.

By default LOLA does not rely on the (possibly misleading) approximation model, but only on the true response. This is useful here since it allows us to consider the model selection results independent of the sample selection settings. I.e., the final distribution of points chosen by LOLA is the same across all runs and model types. This means that any difference in performance between models can not be due to differences in sample distribution. However, in many cases it may be desirable to also include information about the surrogate model itself when choosing potential sample locations. In this case the LOLA algorithm can be combined with one or more other sampling criteria that do depend on model characteristics (for example the *Error*, LRM, and EGO algorithms available in SUMO).

7.10.2 GA settings

The GA is run for a maximum of 15 generations between each sampling iteration (after sampling, the GA continues with the final population of the previous iteration). It terminates if one of the following conditions is satisfied: (1) the maximum number of generations is reached, or (2) 8 generations with no improvement. The size of each deme is set to 15. The migration interval m_i is set to 7, the migration fraction m_f to 0.1 and the migration direction is *both* (copies of the m_f best individuals from island i replace the worst individuals in islands $i-1$ *and* $i+1$). A stochastic uniform selection function was used. Since we want to find the best approximation over the *complete* design space, the fitness of an individual is defined as the root relative square error (RRSE):

$$RRSE(\mathbf{y},\tilde{\mathbf{y}}) = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{7.5}$$

where y_i,$\tilde{y}_i$,$\bar{y}$ are the true, predicted, and mean true response values respectively. Intuitively the *RRSE* indicates how much better an approximation is than the most simple approximation possible (the mean) [25]. In the case of the BH and LE examples no separate validation set is available, instead 20% of the available data is reserved for this purpose (taking care to ensure the validation set is representative of the full data set by maximizing the minimum distance between validation points). Note that we are using a validation set since it is cheap and we have enough data available. In the case where data is scarce we would most likely use the more expensive k-fold cross validation as a fitness measure. This is the case for the EM example.

The remaining parameters are set as follows: $p_m = 0.2, p_c = 0.7, k = 1, p_{swap} = 0.8, el = 1, ES_{max} = 3, ES_{diff} = 0.1, T_{min} = 2$. The random generator seed was set to Matlab's default initial seed.

7.10.3 Termination criteria

In case of adaptive modeling only (no sample selection), the objective is to see what the most accurate model is that can be found in a limited period of time (a typical use case). Thus the required accuracy (target fitness value) is set to 0. For the LE the timeout is set to 180 minutes. For the BH example the timeout is significantly extended to 1200 minutes. Given the high dimensionality, the noise and discontinuities in the input domain it is a hard problem to fit accurately. In this case we are more interested to see how the population would evolve over such an extended period of time.

In case of adaptive sampling, the criteria are: a target accuracy (RRSE) of 0.01, and for the AF example a maximum number of 500 data points is enforced (to see what performance can be reached with a limited sample budget).

7.10.4 Others

Each problem was modeled twice with the heterogeneous evolution algorithm (once with $EP = true$, once with $EP = false$) and once with homogeneous evolution (a single model type run for each model type in the HGA). To smooth out random effects each run was repeated 15 times. This resulted in a total of 516 runs which used up a total of at least 130 days worth of CPU time (excluding initial tests and failed runs). All experiments were run on CalcUA, the cluster available at the University of Antwerp, which consists of 256 Sun Fire V20z nodes (dual AMD Opteron with 4 or 8 GB RAM), running SUSE linux, and Matlab 7.6 R2008a. Due to space considerations, only the results for the *S11* (EE), and *lift* (LE) outputs are considered in this chapter. For all examples the input space is normalized to the interval $[-1, 1]$.

7.11 Discussion

We now discuss the results of each problem separately in the following subsections.

7.11.1 Ackley Function

The composition of the final population for each run is shown in Figure 7.12 for Extinction Prevention (EP) equal to *true* and *EP=false*. The title above each sub figure shows the average and standard deviation over all runs. The first element of each vector corresponds to the first (top) legend entry. The error histogram of the final model of each run on the test data is shown in Figure 7.13. The population evolution for the run that produced the best model in both cases is shown in Figure 7.14. Figure 7.15 then depicts the evolution of the relative error (calculated according to equation 7.6) on the test set as modeling progresses (again in both cases, for the run that produced the best model). The lighter the regions in Figure 7.15, the larger the percentage of test samples that have low relative error (RE) (according to equation 7.6).

$$RE(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{|\mathbf{y} - \tilde{\mathbf{y}}|}{1 + |\mathbf{y}|} \tag{7.6}$$

Finally, a summary of the results for each run is shown in Table 7.1. The table shows the number of samples used ($|X|$), the validation error (VE), the test set error (TE), and the run time for each experiment. All entries are averaged over 15 runs with the standard deviation shown in the adjacent column. The plot of the best model found overall is shown in Figure 7.16.

Regarding the composition of the final population in Figure 7.12, we see that the results are somewhat mixed for *EP=false*. In some runs RBF models perform best, in others LS-SVM models. This is also reflected in the corresponding error histogram plot in Figure 7.13. The quality of the best model found in each run differs considerably between runs. In contrast, for *EP=true*, the results are more clear cut, RBF models dominate in 14 of the 15 runs. This already demonstrates the usefulness of extinction prevention. Due to randomness in the initial population and genetic operators a model type may be driven extinct, unable to return. EP prevents this. In this particular case LS-SVM models generally perform best initially, pushing the RBF models out of the population. However, as more data becomes available (active learning), and as the hyperparameter optimization continues, superior RBF models are discovered and quickly take over the population. This is also nicely shown in Figure 7.14. In both cases the RBF models are driven out of the population around generation 50. Though in the *EP=true* case the RBF models are able to make a re-appearance around generation 100.

Of course nothing prevents this process from recurring. The fact that the optimal solution changes with time is not a disadvantage and should actually be expected since the optimization landscape is dynamic (due to the incremental sampling). Without EP these oscillations are impossible and everything depends on the initial conditions. As a result the danger of converging to a poor local optimum is considerably greater. Given the form of the Ackley function, we should really not be surprised that the RBF models end on top. The different radial basis functions that make up the RBF model (a local model) can be expected to match up quite well with the 'bumps' of the Ackley function.

If we assess the quality of the final models (Figure 7.15) we see that it performs very well. After 500 samples the model has an error smaller than 0.01 on 98% of the test samples. More importantly, these results are consistent as can be seen from the *EP=true* plot in Figure 7.13. Actually, from an application standpoint consistency at this level (accuracy) is more important than consistency in model type selection. Since at the end of the day, from an application perspective, the accuracy of the model is typically most important, not its type.

The natural question that remains, is how do these results compare with simply doing multiple homogeneous evolution (single model type, using the same GA settings) runs, one for each type separately? Those results are shown in Table 7.1. Studying the table we see that the HGA compares favorably. The accuracy of the final models are the essentially the same as those found by the best performing single model type run, while the variance on the results tends to be lower (EP=true). Of course this is paid for by an increase in computation time due to the increased population size of the HGA. Still, the HGA has a factor of 6 larger population size (90 versus 15) but requires only double the running time of the best performing homogeneous run (ANN). Also the total HGA running time is still less than the combined run time of all homogeneous runs.

7.11.2 Kotanchek Function

The composition of the final population and final error histograms for each run are shown in figures 7.17 and 7.18. The population evolution and corresponding error evolution for the best run are shown in figures 7.19 and 7.20. The comparison with homogeneous evolution is shown in Table 7.2.

The Kotanchek function is an interesting example since the GA has to 'discover' that 3 of the 5 variables are irrelevant. Considering the composition of the final population the Kriging functions seem to be able to do this best in the *EP=false* case, with sporadic 'wins' for rational functions. In the *EP=true* case the situation is different, rational functions dominating all 15 runs. The fact that the rational functions succeed in doing this is thanks to a weighting scheme used in the genetic operators and described further in [274].

The usefulness of EP is demonstrated again as well. While the results of the best run for *EP=false* are better than the best run for *EP=true* (less samples), the former is much more a product of chance than the latter (which has lower variance). *EP=true* should still be preferred as it is more robust. Finally, the quality of the final models is excellent in all runs, and the performance and running time of the HGA remains competitive with the single model type runs.

7.11.3 EM Example

The composition of the final population for each run is shown in Figure 7.21, and the associated error histogram in Figure 7.22. The population evolution and corresponding error evolution for the best run are shown in figures 7.23 and 7.24. Table 7.3 summarizes the results and a plot of the best model can be found in Figure 7.25. Note that Table 7.3 shows the cross validation error (*CV*) instead of the validation error.

The results are very clear cut, rational functions dominate in every run, easily reaching the accuracy requirements in about 200 data points (with the *EP=true* runs generally reaching higher accuracies). This is to be expected. The physical behavior of two inductive posts in a rectangular waveguide is well described by a quotient of two differential multinomials (the transfer function) and it is this function that needs to be modeled. Thus it is not surprising that rational functions do well since their form fits the underlying function.

If we compare the HGA runs with the single model type runs we see significant improvements. Interestingly, the HGA runs need roughly 33-25% less sample evaluations to reach the target accuracy, and do so in a fraction of the time (less then 8 minutes vs. an average of 43 minutes for the homogeneous runs). Thus here we have a strong case for the use of the HGA.

7.11.4 LGBB Example

The composition of the final population for each run is shown in Figure 7.26 and the associated error histogram in Figure 7.27. The population evolution of the best run is shown in Figure 7.28. Table 7.4 shows the comparison with the homogeneous runs. A plot of the response can be found in Figure 7.29.

Adaptive sampling was switched off for the LGBB example. The objective was to see what accuracy can be reached and what model type prevails within a fixed time budget. The LGBB example consists of a 3 dimensional data set and unlike the AF and EE examples there are no clues as to which model type is most adequate. Running the heterogeneous evolutionary algorithm it turns out that ANNs give the best fit overall (see Figure 7.26), achieving excellent accuracy. Changing the *EP* setting does not influence this, though the variance is lower for the *EP=true* case.

Within the same time limits the models produced by the HGA are comparable in accuracy to the best performing homogeneous runs, which again demonstrates the usefulness of the HGA.

Interestingly it turns out that the third dimension is negligible, the three slices in Figure 7.29 almost coincide. This was confirmed by using the SUMO model browser to fully explore the response. Thus we can safely conclude that side-slip angle has little or no effect on the lift on re-entry of the LGBB into the atmosphere.

7.11.5 Boston Housing Example

The final example is the Boston Housing data set, adaptive sampling was also switched off. The composition of the final population for each run is shown in Figure 7.30 and the associated error histogram in Figure 7.31. The population evolution of the best run is shown in Figure 7.32. Table 7.5 shows the comparison with the homogeneous runs.

This is a somewhat curious example since it has high dimensionality (13), small support (506 tuples), and the types and ranges of the different inputs parameters vary greatly (e.g., input 4 (CHAS) is a boolean variable that is 1 if the tract borders the river and 0 otherwise while input 5 (NOX) is the nitric oxide concentration). Consequently, any analysis of this data should be preceded by a thorough statistical treatment (feature selection, variance analysis, etc.). We explicitly chose not to do this but take the data as is and treat is as black-box regression problem.

The results are mixed (*see* Figure 7.30), though the HGA runs again outperform the homogeneous runs. (LS-)SVM and ANN models seem to be preferred over Kriging and RBF models but there is no evidence to distinguish between the models any further. Striking, though, is that about half of the final population consists of ensembles and that most of these ensembles turn out to be {ANN, RBF} pairs or multiple ANNs. Figure 7.34 shows the evolution of the composition of the best performing ensemble. The popularity of ensembles in this case is in line with the authors' previous experiences. When the individual model types are having trouble to fit a difficult response with none really performing much better than the other, hybrids (ensembles) tend to do well since they can produce more complicated responses. It is a signal that none of the included model types are really fit for the approximation problem.

Also striking (and interesting) are the oscillations in the population evolution (*see* Figure 7.32, or Figure 7.33 for a more marked example). It turns out that every run shows these oscillations between ensembles and one or two other model types. Interestingly these 'spikes' occur every 10 or 7 generations. It remains unclear to the authors how these oscillations may be explained. This is an issue that is being investigated in more detail.

7.12 Summary

In summary the results for the different test problems are very promising and in line with previous results [267, 273, 370, 371]. The results show a consensus about which model type to use in all test cases (ignoring the BH example for the moment). In the case the consensus is not absolute (e.g., the *EP=true* run for the AF in Figure 7.12) the final model accuracies are essentially the same thus this is not really a problem from an application standpoint. More important is that the target accuracy has been reached and that all model types have been given a fair chance without having to resort to a brute force approach.

In general we found the algorithm to be quite consistent across many runs. When variation does show up in the model selection results it typically is because two or more model types can fit the data equally well with only a minor difference in accuracy. This means that the GA may alternate between the different local optima, giving different model selection results, but still reaching the targets. The other reason is if the data is simply too difficult to fit using the methods included in the evolution. In this case ensembles may tend to do well. The BH example seems indicative of this situation.

It is important to remind the reader, though, that the overall performance of the HGA will of course depend on the quality of the model types themselves, and more importantly, on the quality of the creation function and genetic operators (and adequacy of the chosen representation). Good results have already been obtained with the current implementations though there is still room for improvement. The final surrogates are not neccessarily as refined as an expert in any one type would wish. Luckily, an advantage of the HGA based approach is that since the general algorithm is now in place, it becomes possible to focus on such improvements without requiring changes to the HGA itself. Specific improvements (e.g., devised by an expert in a certain model type) are straightforward to integrate into the existing genetic operators allowing an accumulation knowledge that will improve the overall quality of the models produced by the HGA.

A next step is to further increase the number of test problems and, more importantly, investigate the influence of the different parameters involved. In this respect the migration interval and migration topology parameters are particularly important since they determine how the different model types interact. For example, if the migration interval is set too high, each deme will produce high quality models (most of the time is spent optimizing the parameters of a single model type) but there will have been very little competition between models. If it is set too low, the converse is true. More research is needed to to better understand this balance and investigate the impact of genetic drift.

7.13 Application in Optimization

In addition to the global modeling use case, initial tests have been done linking the HGA described here with the infill framework described in section 4.6. This allows for automatic model type switching during optimization (any model type that supports prediction variance can be used) and may be beneficial for computationally expensive codes. We briefly discuss this case in this section.

7.13.1 Background

Starting from an initial approximation of the design space, an infill criterion identifies new samples of interest (infill or update points) to update the approximation model. It is crucial in global SBO to strike a correct balance between exploration[5] and exploitation[6]. A well-known infill criterion that is able to effectively solve this trade-off is Expected Improvement (EI), popularized by Jones et al. [89, 378] in the Efficient Global Optimization (EGO) algorithm.

While expected improvement is proven to be an efficient figure of merit, the quality of the surrogate model is still arguably the most important factor in the optimization process. The surrogate model of choice in the EGO algorithm is the kriging model as it provides the prediction variance needed by expected improvement. However other surrogate models such as Support Vector Machines (SVM), pure Gaussian Processes (GP), etc. are possible and may have improved accuracy on some problems. Unfortunately it is rarely possible to choose an appropriate surrogate model a priori. Typically, the behavior of the objective function is poorly understood or even unknown. In this case the automatic model type selection algorithm can be useful.

7.13.2 Application

To illustrate how the HGA may be used in an infill optimization framework we take a structural dynamics problem. The problem is the optimal design of a two-dimensional truss, constructed by 42 Euler-Bernoulli beams and a simplification of a truss type

[5]enhancing the general accuracy of the surrogate model
[6]enhancing the accuracy of the surrogate model solely in the region of the (current) optimum

typically used in satellites [379]. More details are given in the next chapter in section 9.4.2.

The SUMO Toolbox is utilized to optimize the truss structure using the original expected improvement function as defined in section 4.6. The expected improvement function is optimized using the DIviding RECTangles (DIRECT) algorithm of Jones et al. [380] to determine the next sample point to evaluate. Whenever the DIRECT algorithm is unable to obtain a unique sample a fallback criterion is optimized. This fallback criterion represents the Mean Squared Error (MSE) between the current best surrogate model and all previous surrogate models that have been kept in the history. In effect, identifying locations in the domain where the prediction of surrogate models disagree on.

The aforementioned configuration is reproduced three times with different surrogate modelling strategies. The first two cases configure kriging as surrogate model of choice. More precisely, one time hyperparameters are obtained through maximum likelihood estimation (MLE) using SQPLab [381] (utilizing likelihood derivative information). In the other, second, run, the hyperparameters of the kriging model are identified by Matlab's Genetic Algorithm (GA) toolbox guided by 5-fold cross validation. In the last, and final, configuration the surrogate model is produced by the evolutionary model type selection algorithm as described in this chapter.

Note, as the EI approach is used for optimization only model types that support a point-wise error estimation (prediction variance) can be included in the EMS method. Thus, the surrogate model types that compete in the evolution are kriging models, Radial Basis Functions (RBF) and LS-SVMs (using [269]).

Each of the three different use cases is allowed to run for 350 samples, in other words, the optimization process halts after 350 calls to the simulator. Finally, the tests are repeated 20 times to smooth out random effects.

7.13.3 Results

The average minimum function value at each intermediate dataset is shown in figure 7.10. It is seen that the EMS algorithm is consistently better than the other two surrogate modeling strategies, though the difference is quite small. Note that the evolution of the minimum function value of the EMS algorithm is more smooth than, for instance, kriging (MLE). For the latter, the function value decreases in a stepwise way, while for the EMS algorithm other surrogate models than kriging are used whenever they are better in approximating the current set of samples, thus, a better function value is found more often.

More compelling is the amount of variance on the final optima. A boxplot of the final function value for the three surrogate modeling strategy is seen in Figure 7.11. Kriging (GA) and the EMS algorithm are substantially more stable than using kriging (MLE), where the θ parameters are optimized using the likelihood. The EMS

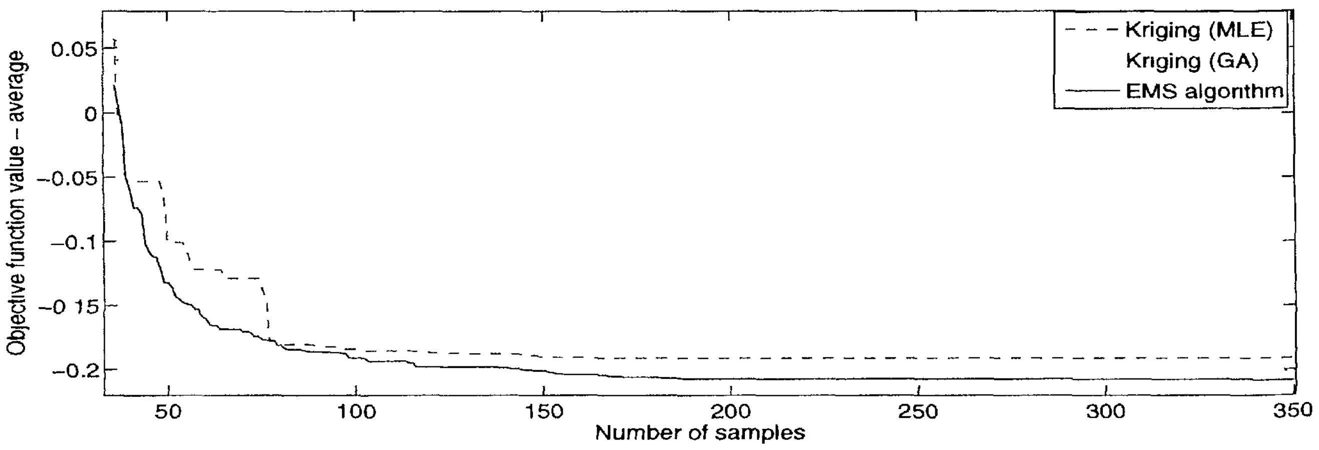


Figure 7.10: Evolution of the number of samples versus the minimum value

algorithm is also an improvement over kriging (GA) as the quantiles lie closer to the smallest function value found.

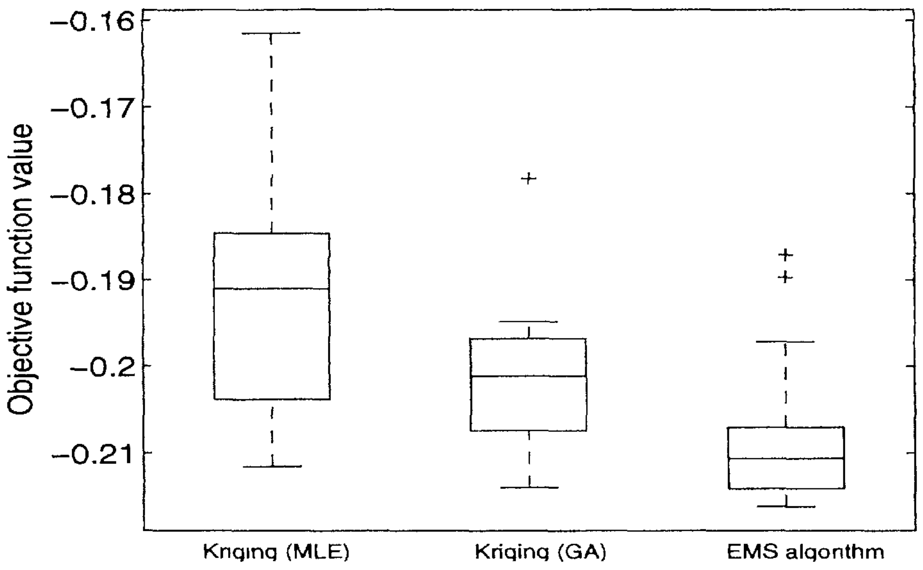


Figure 7.11: Box-and-whiskers plot of the final optimum

The final best surrogate model is almost always an ensemble model, with only 1 case out of 20 where a sole RBF model gives the best accuracy. Furthermore, the final ensemble models consist mostly of RBF models. The share of SVM models and kriging models in those ensembles is identical.

7.14 Conclusion

A recurring problem in surrogate modeling is selecting the most adequate surrogate model type and associated complexity. In this chapter we explored an approach based on the evolutionary migration model that can help tackle this problem in an automatic way if little information is known about the true response behavior and there are no a priori model type requirements. In addition, we have illustrated the usefulness of extinction prevention and ensemble based recombination. Extinction prevention is a straightforward algorithm that prevents a species from disappearing from the gene pool at the expense of a minor cost (keeping 2 extra individuals per species 'alive'). As a result, the optimal solution is able to change with time, making for a more flexible and adaptive system which, as demonstrated in the different examples, gives better and more consistent results.

Future work consists of investigating the oscillations in the BH example, exploring different GA parameter values (role of the migration frequency, migration topology, etc.), incorporating more model types, and more advanced ensemble methods (e.g., stronger constraints on ensemble composition). As mentioned above, improvements to the genetic operators are ongoing in order to get more out of each model type. The utility of adding a penalty to the fitness function proportional to the model complexity and/or training time will also be investigated. Furthermore, we have been experimenting with sampling strategies that vary dynamically depending on the remaining sample budget and quality & type of surrogate currently used in the modeling process. The idea is to work towards an optimal interplay between sampling and modeling. E.g., initially the focus should be on exploration of the design space while, as accuracy of the models improves, the focus should shift towards refining the model in places where it is uncertain and ensuring the optima it exhibits are really true optima. Likewise, we are experimenting with dynamic model selection criteria. For example, if only little data is available cross validation type measures may be unreliable and it makes little sense enforcing problem specific constraints (e.g., the model response should be bounded between given bounds). However, when the data density is sufficiently high the opposite will be true. Thus there seems to be some intuition advocating the use of annealing type strategies.

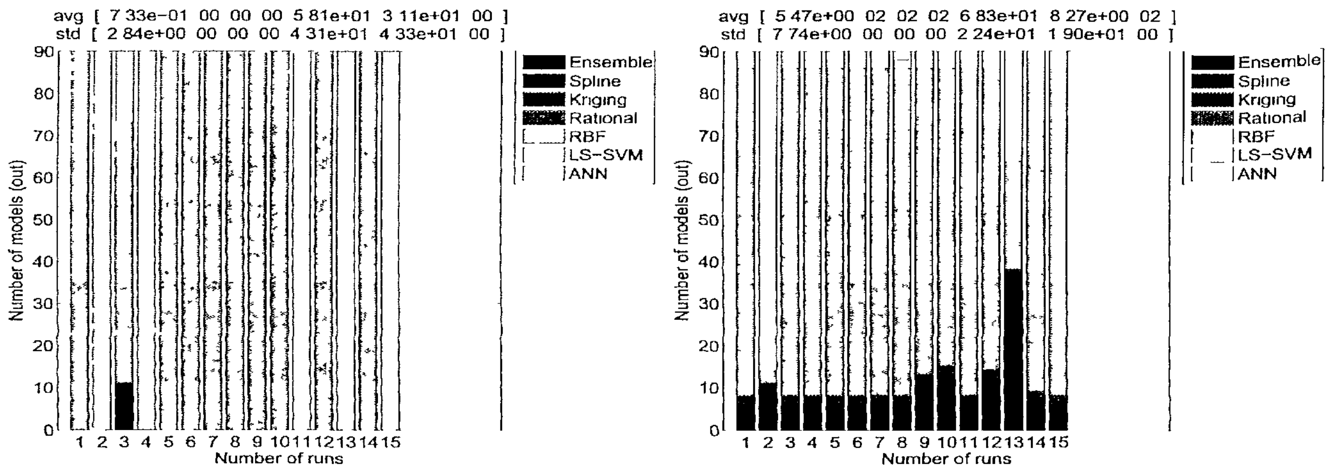


Figure 7.12: AF: Composition of the final population (Left. EP=false, *Right:* EP=true*)*

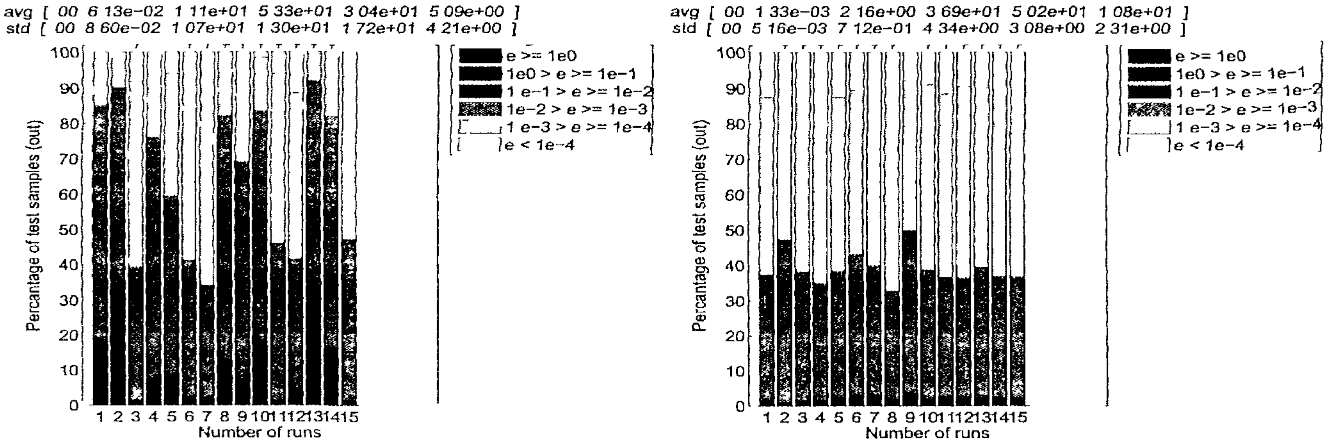


Figure 7.13: AF: Error histogram of the final best model in each run (Left: EP=false, *Right:* EP=true*)*

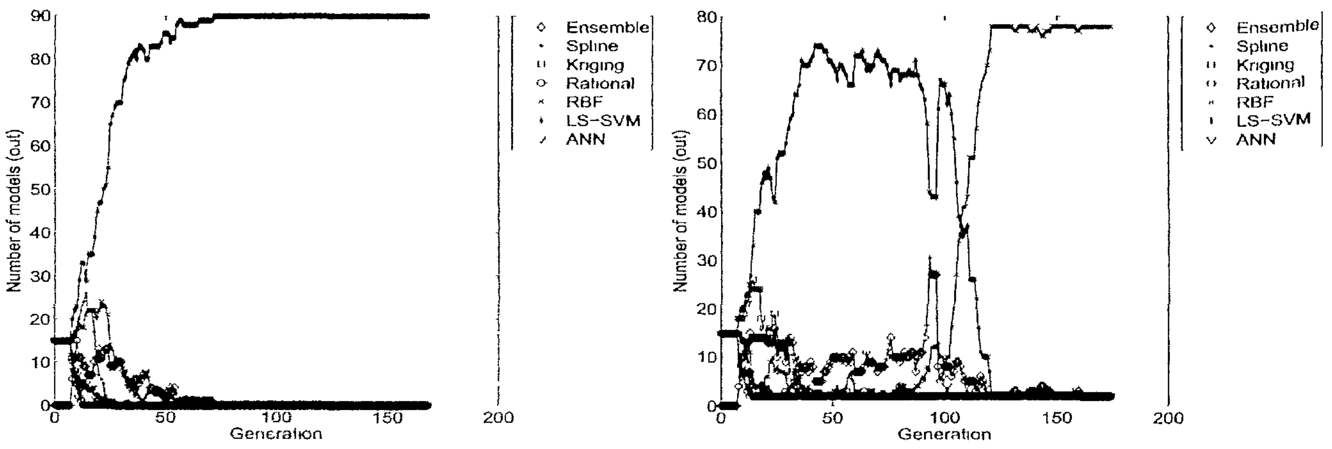


Figure 7.14: AF: Population evolution of the best run (Left: EP=false, *Right:* EP=true*)*

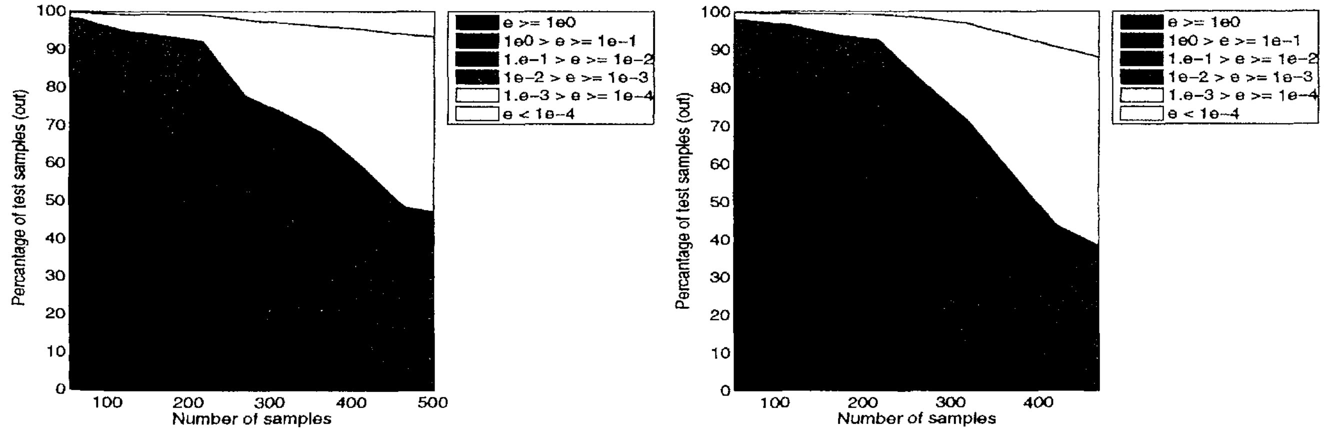


Figure 7.15: AF: Error evolution of the best run (Left: EP=false, *Right:* EP=true*)*

Method	$\|X\|$	σ	$\overline{VE}_{RRSE}$	σ	$\overline{TE}_{RRSE}$	σ	$\overline{time}$ *(min)*	σ
ANN	4.97E+02	1.75E+01	1.30E-02	3.42E-03	1.29E-02	3.45E-03	1.81E+02	3.17E+01
Kriging	5.26E+02	1.19E+01	2.01E-02	4.26E-03	2.03E-02	4.52E-03	7.25E+01	9.19E+00
LS-SVM	5.20E+02	1.20E+01	1.36E-02	2.26E-03	1.37E-02	2.31E-03	3.99E+01	3.27E+00
Rational	5.17E+02	1.02E+01	1.88E-01	5.85E-02	1.86E-01	5.93E-02	2.03E+01	2.22E+00
RBF	5.19E+02	1.54E+01	1.32E-02	2.36E-03	1.32E-02	2.31E-03	4.37E+01	4.13E+00
Splines	5.30E+02	1.36E+01	2.47E-02	5.66E-03	2.42E-02	5.40E-03	4.29E+01	4.98E+00
$HGA_{EP=false}$	5.05E+02	6.51E+00	3.14E-02	1.77E-02	3.09E-02	1.71E-02	2.36E+02	1.56E+02
$HGA_{EP=true}$	5.04E+02	0.00E+00	1.34E-02	1.93E-03	1.36E-02	1.89E-03	3.69E+02	6.15E+01

Table 7.1: AF: Comparison with homogeneous evolution

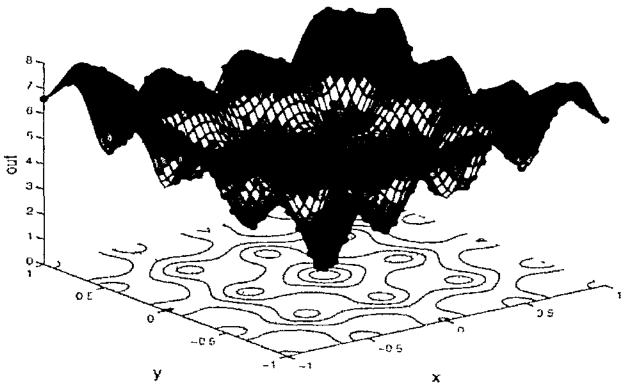


Figure 7.16: AF: normalized plot of the best model overall ($HGA_{EP=true}$)

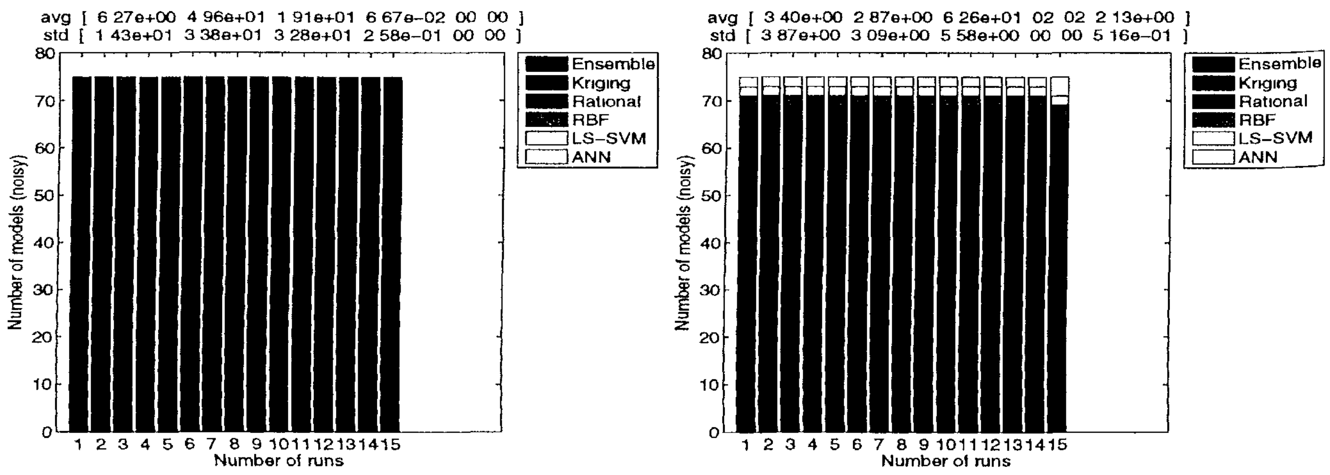


Figure 7.17: KF: Composition of the final population (Left: EP=false, *Right:* EP=true*)*

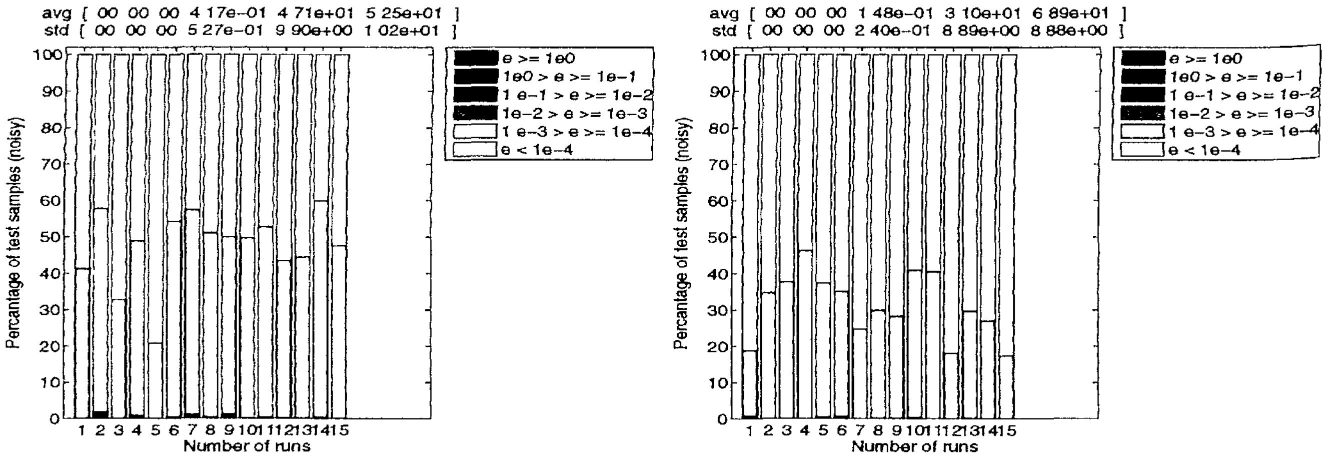


Figure 7.18: KF: Error histogram of the final best model in each run (Left: EP=false, *Right:* EP=true*)*

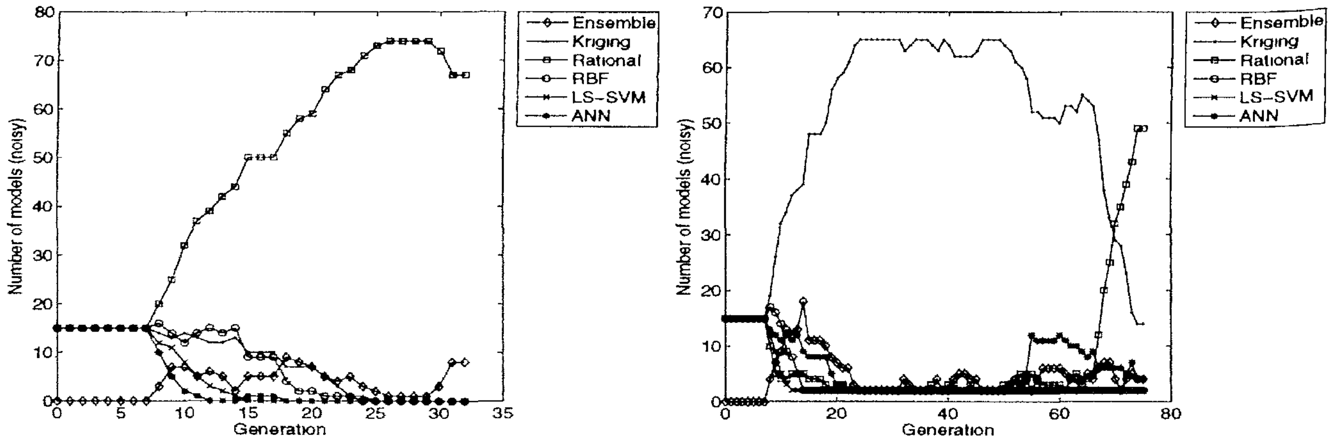


Figure 7.19: KF: Population evolution of the best run (Left: EP=false, *Right:* EP=true*)*

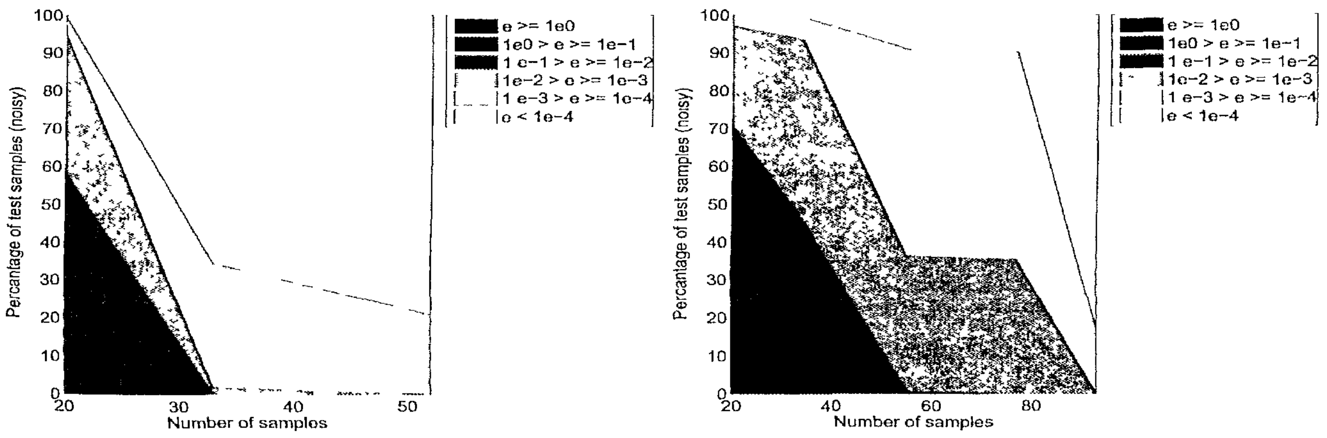


Figure 7.20: KF: Error evolution of the best run (Left: EP=false, *Right:* EP=true*)*

Method	$\|X\|$	σ	$\overline{VE}_{RRSE}$	σ	$\overline{TE}_{RRSE}$	σ	$\overline{time}$ *(min)*	σ
ANN	1.41E+02	2.20E+01	8.10E-04	1.61E-04	8.45E-04	1.67E-04	6.05E+01	1.25E+01
Kriging	5.20E+02	0.00E+00	1.98E-03	3.54E-04	2.02E-03	3.61E-04	1.17E+02	6.22E+01
LS-SVM	5.10E+02	2.84E+00	1.18E-01	4.96E-03	1.19E-01	5.30E-03	5.67E+01	3.57E+00
Rational	1.47E+02	9.14E+01	5.88E-04	2.11E-04	6.27E-04	2.46E-04	1.19E+01	7.54E+00
RBF	5.20E+02	0.00E+00	1.07E-01	3.62E-03	1.08E-01	4.41E-03	9.18E+01	2.15E+01
$HGA_{LP\ false}$	3.05E+02	2.07E+02	1.07E-03	3.39E-04	1.08E-03	3.25E-04	3.03E+02	2.65E+02
$HGA_{FP\ true}$	6.26E+01	1.48E+01	7.00E-04	2.28E-04	7.09E-04	2.02E-04	5.49E+01	3.53E+01

Table 7.2: KF: Comparison with homogeneous evolution

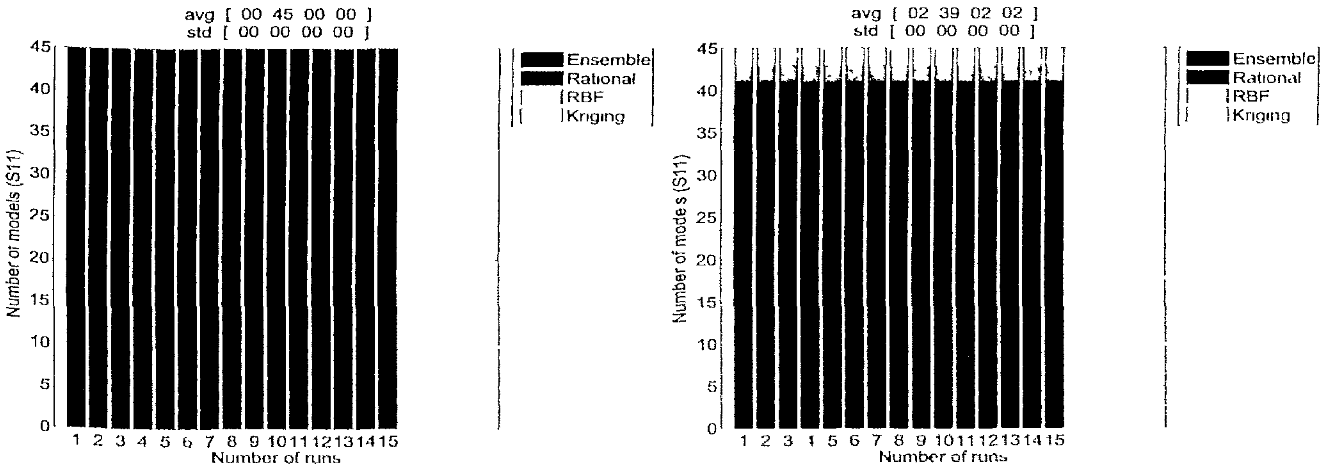
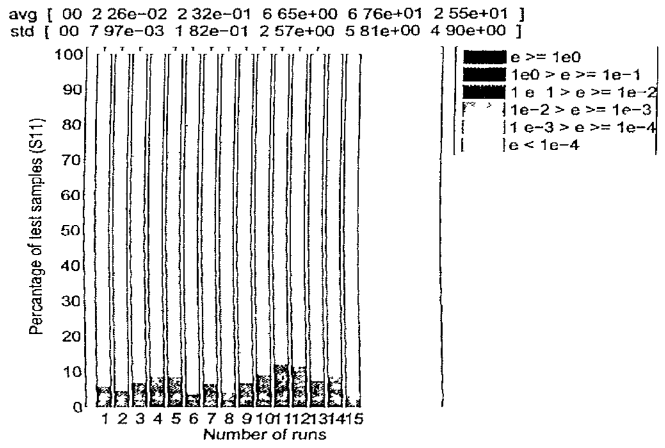


Figure 7.21: EE: Composition of the final population (Left: EP=false, *Right:* EP=true*)*



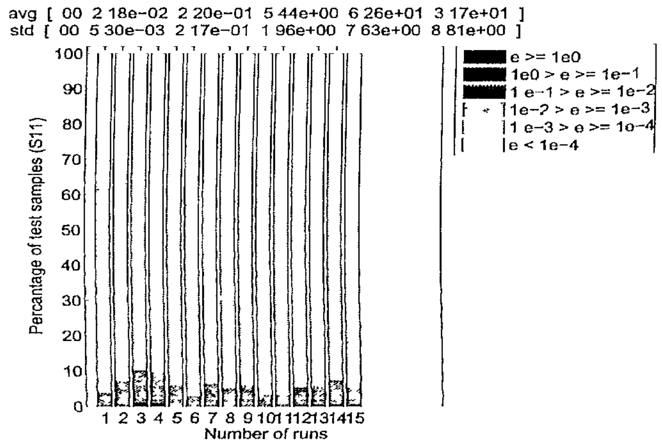
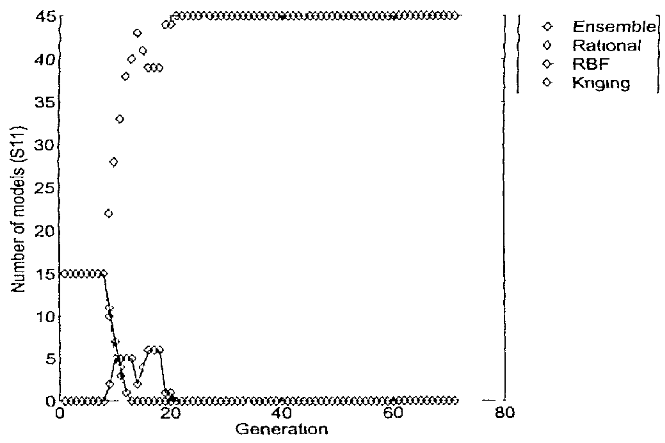


Figure 7.22: EE: Error histogram of the final best model in each run (Left: EP=false, *Right:* EP=true*)*



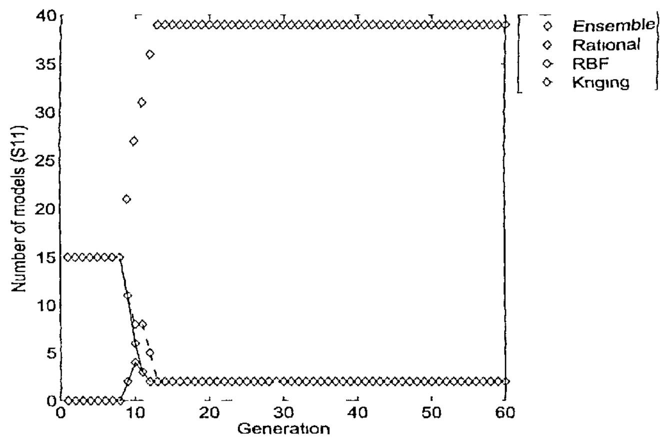
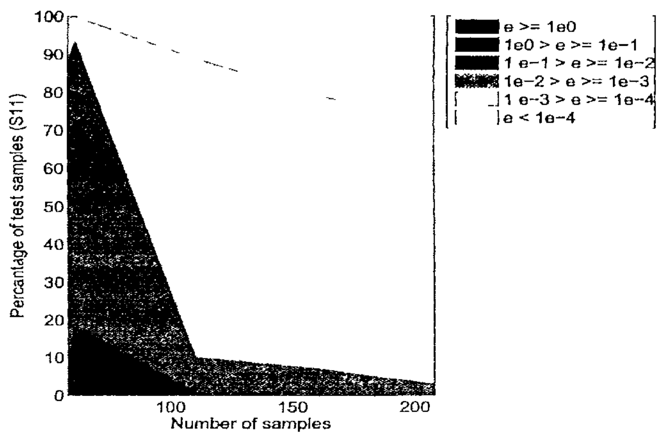


Figure 7.23: EE: Population evolution of the best run (Left: EP=false, *Right:* EP=true*)*



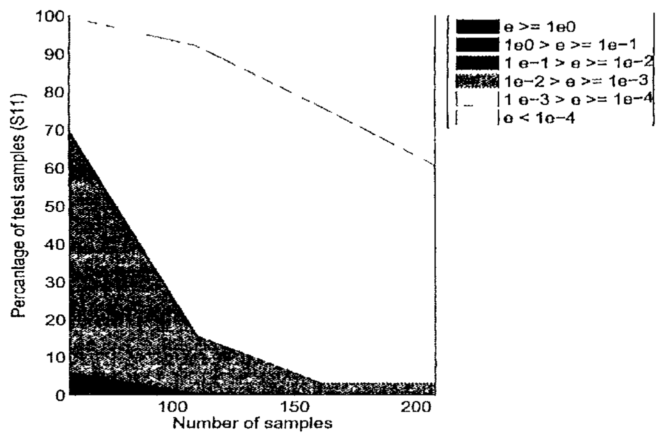


Figure 7.24: EE: Error evolution of the best run (Left: EP=false, *Right:* EP=true*)*

Method	$\lvert X \rvert$	σ	$\overline{CV}_{RRSE}$	σ	$\overline{TE}_{RRSE}$	σ	$\overline{time}$ *(min)*	σ
Kriging	7.98E+02	1.13E+02	8.54E-03	7.92E-04	1.88E-02	2.83E-03	7.20E+01	2.78E+01
Rational	8.14E+02	2.31E+02	1.15E-02	4.12E-03	1.70E-02	4.97E-03	4.04E+01	1.91E+01
RBF	6.08E+02	4.22E+01	8.12E-03	6.33E-04	1.55E-02	2.40E-03	1.71E+01	2.15E+00
$HGA_{EP=false}$	1.88E+02	2.53E+01	6.29E-03	1.77E-03	3.51E-02	4.57E-02	7.90E+00	1.44E+00
$HGA_{EP=true}$	1.98E+02	2.07E+01	6.73E-03	1.45E-03	2.22E-02	1.14E-02	7.72E+00	1.07E+00

Table 7.3: EE: Comparison with homogeneous evolution

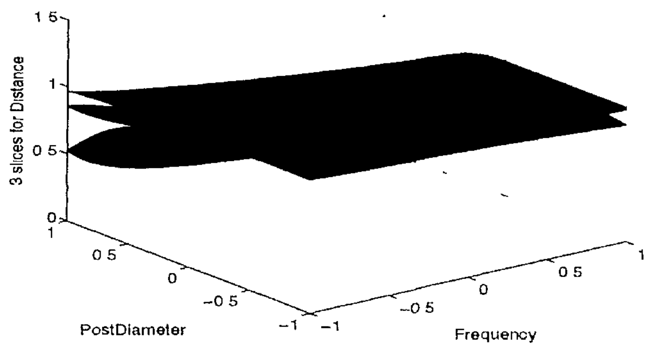


Figure 7.25: EE: normalized plot of $|S_{11}|$ of the best model overall ($HGA_{EP=true}$, 3 slices for Distance)

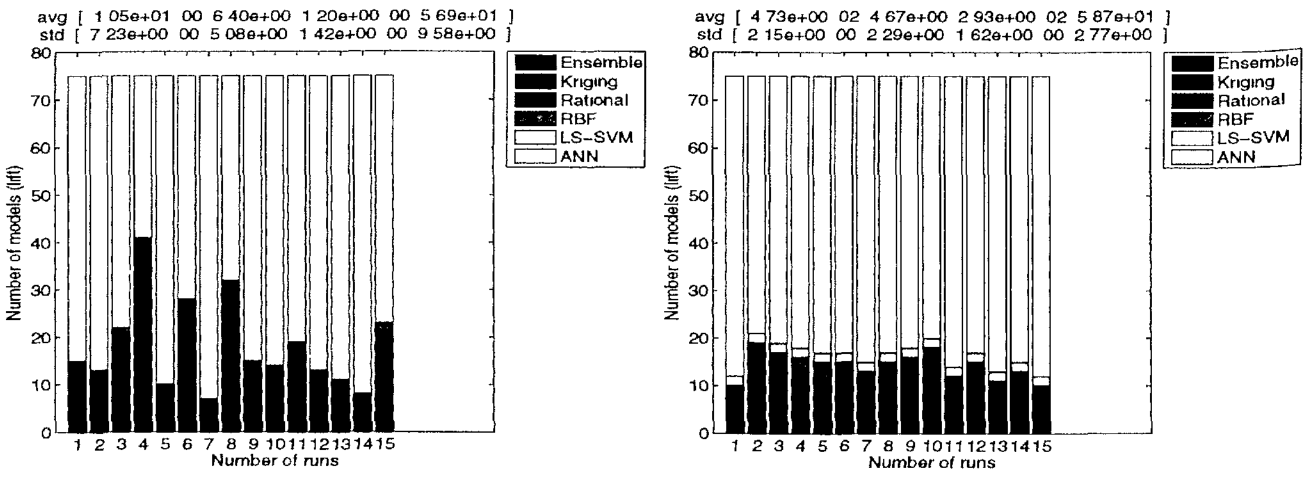


Figure 7.26: LE: Composition of the final population (Left: EP=false, *Right:* EP=true*)*

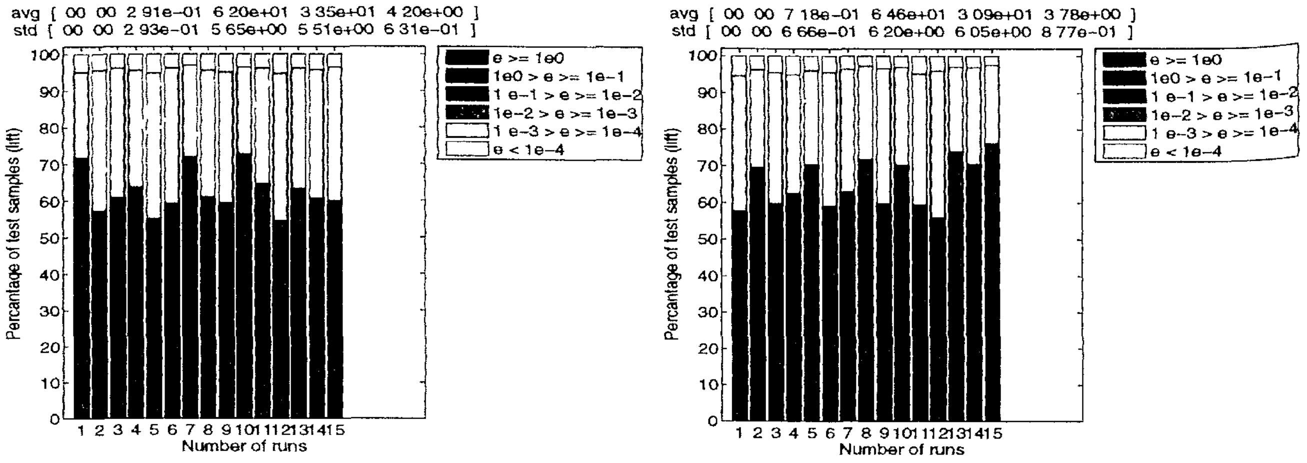


Figure 7.27: LE: Error histogram of the final best model in each run (Left: EP=false, *Right:* EP=true*)*

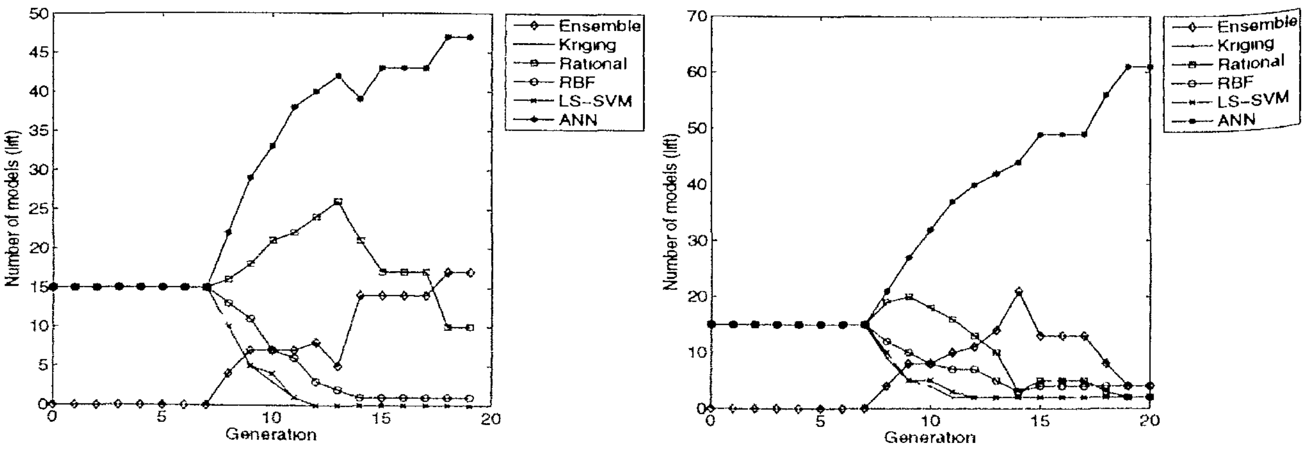


Figure 7.28: LE: Population evolution of the best run (Left: EP=false, *Right:* EP=true*)*

Method	$\|X\|$	VE_{RRSE}^{-}	σ	*time (min)*
ANN	7.800E+02	7.47E-003	5.60E-004	3.00E+002
Kriging	7.800E+02	1.08E-001	1.96E-002	3.00E+002
LS-SVM	7.800E+02	1.40E-001	4.09E-005	3 00E+002
Rational	7.800E+02	5.20E-002	3.02E-004	3.00E+002
RBF	7.800E+02	7.34E-002	3.53E-008	3.00E+002
$HGA_{EP=false}$	7.800E+02	7.68E-003	4.38E-004	3.00E+002
$HGA_{EP=true}$	7 800E+02	7.59E-003	4.26E-004	3.00E+002

Table 7.4: LE: Comparison with homogeneous evolution

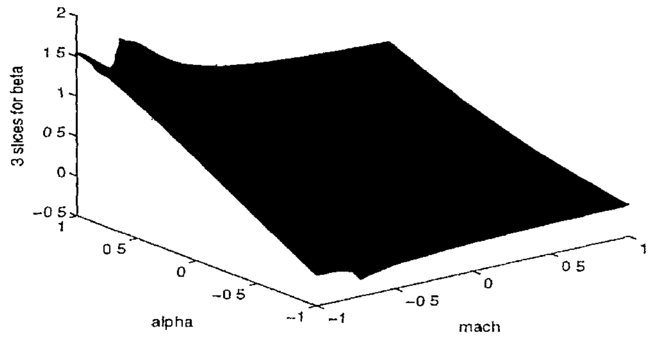


Figure 7.29: LE: normalized lot of the best model overall ($HGA_{EP=true}$, 3 slices for beta*)*

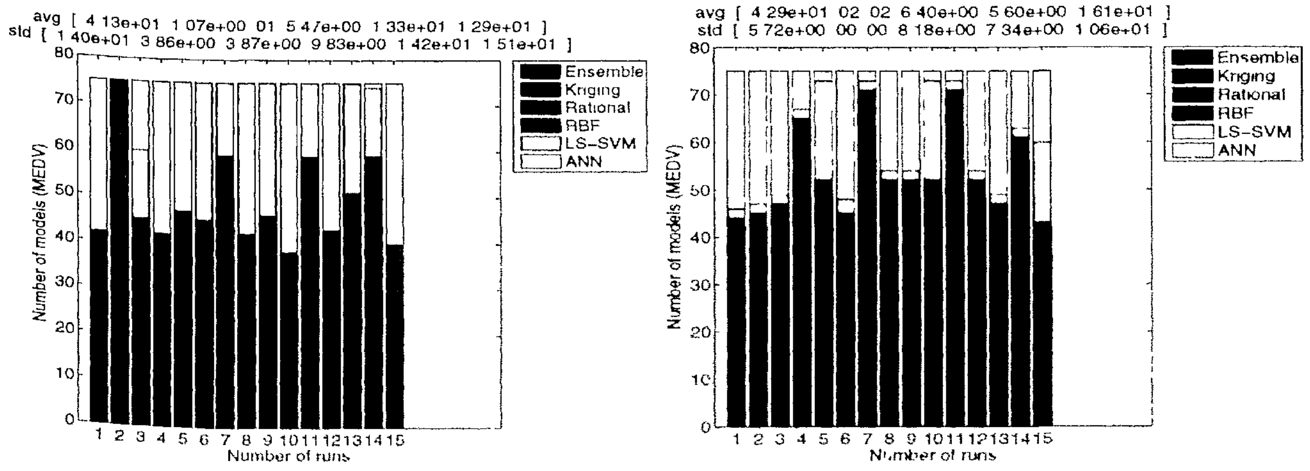


Figure 7.30: BH: Composition of the final population (Left: EP=false, *Right:* EP=true*)*

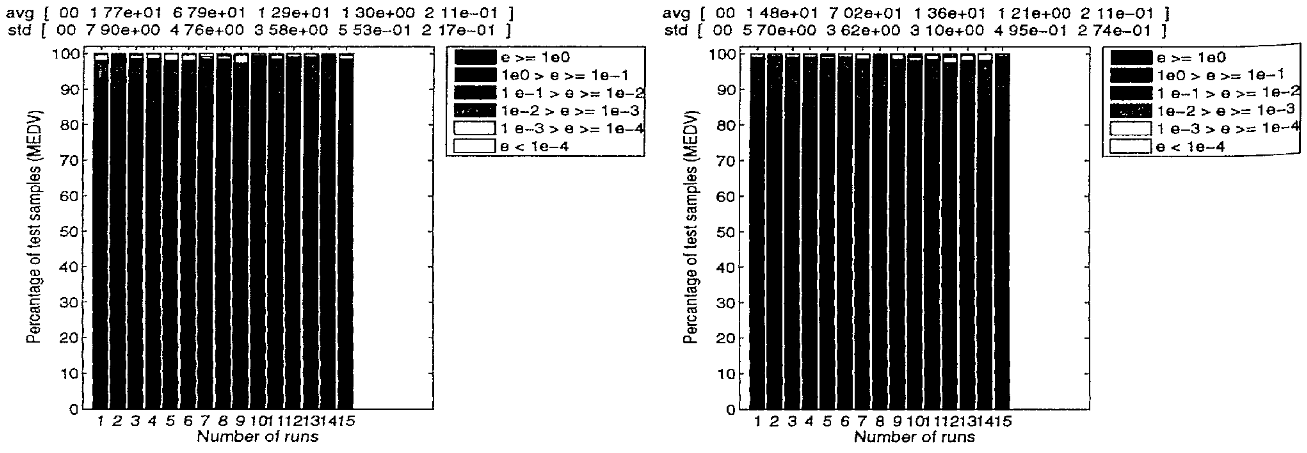


Figure 7.31: BH: Error histogram of the final best model in each run (Left: EP=false, *Right:* EP=true*)*

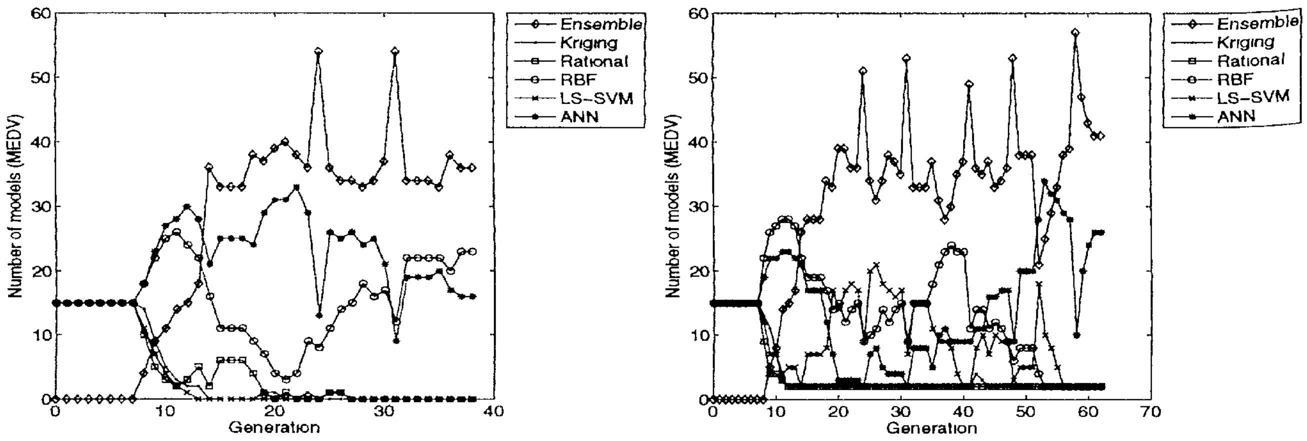


Figure 7.32: BH: Population evolution of the best run (Left: EP=false, *Right:* EP=true*)*

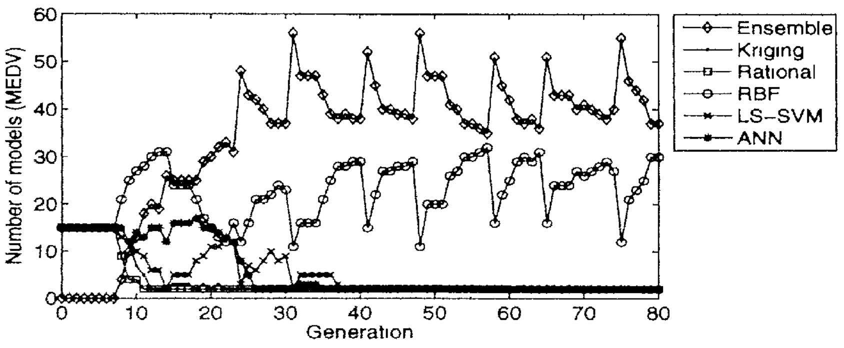


Figure 7.33: BH: Oscillations in the population evolution ($HGA_{EP=true}$)

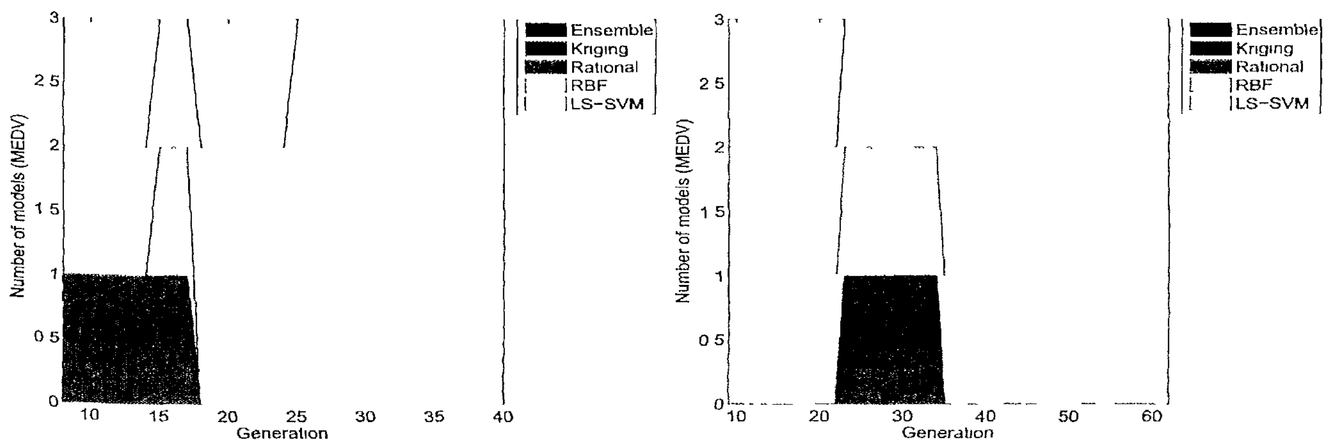


Figure 7.34: BH: Composition of the best performing ensemble of the best run (Left: EP=false, *Right:* EP=true*)*

Method	$\|X\|$	$\overline{VE}_{RRSE}$	σ	$\overline{time}$ *(min)*
ANN	5.060E+02	2.985E-01	8.962E-03	1.200E+03
Kriging	5.060E+02	3.448E-01	1.077E-02	1.200E+03
LS-SVM	5.060E+02	3.421E-01	6.012E-06	1.200E+03
Rational	5.060E+02	5.006E-01	4.296E-02	1.200E+03
RBF	5.060E+02	1.228E+15	2.104E+15	1.200E+03
$HGA_{EP\ false}$	5.060E+02	2.764E-01	2.768E-02	1.200E+03
$HGA_{EP\ true}$	5.060E+02	2.735E-01	1.372E-02	1.200E+03

Table 7.5: BH: Comparison with homogeneous evolution

8 Multi-Objective Modeling

I can teach anybody how to get what they want out of life. The problem is that I can't find anybody who can tell me what they want.

– Mark Twain

8.1 Introduction

As articulated a number of times in this dissertation, arguably the most crucial aspect of the surrogate modeling process is the assesment and estimation of the surrogate model quality. This is also referred to as model validation, which Kleijnen and Sargent [185] define as the "...*verification that a model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model.*". The *"intended application of the model"* part is important. It is impossible to construct a proper model if the intended application, use, and accuracy requirements are poorly defined. Therefore the very first step of the surrogate modeling excercise is to clearly define the desired characteristics that the model should exhibit (smooth versus non-smooth, interpolation versus regression, evaluation speed, data scalability, accuracy bounds, etc.). This is the responsibility of the domain expert. Once these characteristics are known they can be used to drive hyperparameter optimization process. Typically this optimization process is formulated in a single objective way. Models are generated according to a single objective (accuracy). However, this requires an engineer to determine a single accuracy target and measure upfront, which is hard to do if the behavior of the response is unknown. In this chapter we investigate how a multi-objective approach can be used to alleviate this problem and the related problem of multi-response modeling.

8.2 The 5 percent problem

Though previous experience recommends that a metamodel with normalized RMSE (RMSE divided by the sample range of responses) less than 5% and normalized MAX (MAX divided by the samples range of responses) less than 10% is acceptable for design space exploration at early design stages, there is no rigorously-defined guidance on model selection. Currently, with RMSE and MAX we cannot tell "how accurate" one metamodel is, and whether it meets the requirement of designers; what we can do is only to compare the accuracy of different models. [9]

Recall from section 3.2 that the model parameter optimization process is driven by a criterion ξ, where ξ constitutes three parts:

$$\xi = (\Lambda, \varepsilon, \tau) \tag{8.1}$$

with Λ the model performance estimator, ε the error (or loss) function used, and τ the target value required by the user. In the simple case where Λ is just the in-sample error, the model selection problem simplifies to finding the optimal model type $t^* \in T$ and hyperparameter assignment $\theta^* \in \Theta$ such that

$$\varepsilon(\tilde{f}_{t^*,\theta^*}(X_t), f(X_t)) \leqslant \tau \tag{8.2}$$

The crucial problem is identifying suitable implementations for Λ, ε and a target value for τ. The three are closely linked but only the Λ-selection problem has been extensively studied theoretically [171, 382] and empirically [170, 180]. The selection of ε and τ is less appreciated and often overlooked, but equally important [383, 384]. Selecting an error function ε and required target accuracy τ is difficult since it requires knowledge of the structure of the response and a full understanding of what the performance metric Λ actually measures. Failure to do so leads to misinterpretation, inappropriate application, ultimately resulting in a trial and error model generating procedure.

This brings us to, what we have termed, *"The 5 percent problem"*. The phrase stems from an application where an engineer needed a replacement metamodel. When asked what model accuracy was required the answer was simply 5 percent. While this may seem like a straightforward, objective requirement, enforcing it in practice turned out to be difficult. The reason is twofold and is detailed below.

8.2.1 Choice of error function

First, there is the choice of the error function ε. Roughly speaking there are two categories of error functions: absolute and relative.

8.2.1.1 Absolute errors

Absolute errors (e.g., Average Absolute Error (AAE), Mean Squared Error (MSE), etc.) are often undesirable in an application context since they are not unit-free and depend on the specific prediction value of the response. On the other hand they are very popular in machine learning settings but not always rightly used. A good example is the Root Mean Square Error (RMSE), by far the most popular error function:

$$RMSE(\mathbf{y},\tilde{\mathbf{y}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2} \tag{8.3}$$

where $\mathbf{y},\tilde{\mathbf{y}}$ are the real and predicted response values respectively. The main advantage of the RMSE is that it is the best finite-sample approximation of the standard error $\sqrt{E[\mathbf{y}-\tilde{\mathbf{y}}]}$ and standard deviation (in the case of an unbiased model) [383]. However, its use is not recommended since it penalizes large errors too severely while virtually ignoring small errors. Such an error function is called pessimistic. Also it is unintuitive to interpret. The RMSE is often interpreted as the true arithmetic average euclidean distance between the prediction $\tilde{\mathbf{y}}$ and the true value $\mathbf{y}$. This is however not the case, the RMSE is really one $\sqrt{n}$-th of this value and thus has no simple geometrical interpretation whatsoever. A better solution would be to use the Average Euclidean Error (AEE) as proposed by [383]

$$AEE(\mathbf{y},\tilde{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(y_i - \tilde{y}_i)^2} \tag{8.4}$$

However, while the AEE enjoys many desirable properties, it still suffers from outliers (i.e., it is also pessimistic, though less than the RMSE). In cases where this is a problem alternative functions like the Geometric Average Error (GAE) and the Harmonic Average Error (HAE) [383] can be more useful.

$$GAE(\mathbf{y},\tilde{\mathbf{y}}) = \left(\prod_{i=1}^{n}\sqrt{(y_i - \tilde{y}_i)^2}\right)^{\frac{1}{n}} \tag{8.5}$$

$$\begin{aligned} HAE(\mathbf{y},\tilde{\mathbf{y}}) &= \left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\sqrt{(y_i - \tilde{y}_i)^2}}\right)^{-1} \\ &= \frac{n}{\frac{1}{\sqrt{(y_1-\tilde{y}_1)^2}} + \ldots + \frac{1}{\sqrt{(y_n-\tilde{y}_n)^2}}} \end{aligned} \tag{8.6}$$

In contrast to the RMSE and AEE, the HAE is an optimistic error function since it is dominated by the small error terms. The HAE can be appropriate if the error fluctuates greatly over different runs. This property may be useful in the context of k-fold cross validation with relatively few samples. The GAE, on the other hand, is a balanced error function that suffers much less from extremes (large or small). The GAE, however, has as a disadvantage that if the error is zero in a single point, the overall error is also zero. This is of course may not be desirable.

Many more absolute error variants exist and we do not intend to give an exhaustive overview. Rather we wish to illustrate that each function brings its own tradeoffs and interpretation depending on how the absolute differences $|y_i - \tilde{y}_i|$ are aggregated. In general though, absolute error criteria are not ideally suited for performance estimation of an approximation model due to their context dependence (i.e., τ is hard to specify up front and depends on the units used).

8.2.1.2 Relative errors

Thus engineers typically prefer relative or percentage errors, e.g., 5%. A figure of 5% implies some kind of global averaged relative error, but there are different ways to calculate relative errors (depending on what reference and aggregation function is used): Average Relative Error (ARE), Maximum Relative Error (MRE), Relative Squared Error (RSE), Root Relative Square Error (RRSE), Relative Absolute Error I (RAEI), Relative Absolute Error II (RAEII), Root Mean Square Relative Error (RMSRE), etc. [25, 383].

A natural solution is to take the most intuitive error function, the ARE:

$$ARE(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \tilde{y}_i|}{|y_i|} \tag{8.7}$$

By taking the true value as a reference the ARE results in an intuitively understandable number. Multiplied by 100 it results in a natural percentage. However, taking the geometric or harmonic mean (resulting in the Geometric average Relative Error (GRE) and Harmonic average Relative Error (HRE) respectively) instead of the simple average can also be interpreted as a global percentage error. But since ARE, GRE, and HRE treat different types of errors differently (e.g., ARE is more sensitive to large errors than GRE) care should be taken when interpreting a figure like 5%. In addition, the "%" suffix is sometimes also used when using, for example, RRSE (see below). This, however, should be avoided since it is confusing.

The problem with the ARE is that care must be taken in its interpretation when the true function values y_i are small or, in the extreme but not unlikely case, zero. Since then the error tends to infinity, giving a biased result. What is sometimes done to combat this is add one (+1) to the denominator to prevent division by zero. While this solves the numerical issue, the resulting error is an absolute-relative hybrid and becomes impossible to interpret. A different solution is to scale or translate the response to eliminate small absolute values (e.g., as proposed in [198]). However, the best scale factor is not always obvious and shifting the response can introduce its own problems. For example, figure 8.1 illustrates how a simple shifting of the response (+1000) can drastically improve the ARE value (3 orders of magnitude) for exactly the same model parameters (error measured in the samples).

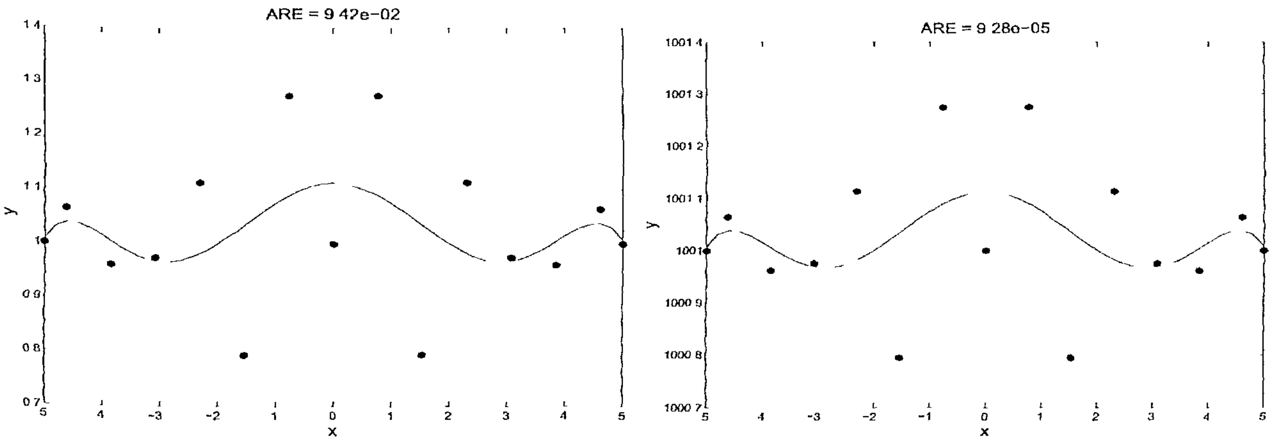


Figure 8.1: Influence of shifting the response on the Average Relative Error

Additionally, there is the well known issue of averaging the errors (relative or absolute). Since the mean is not robust it is easily skewed by outliers and can mask local errors that may be deemed problematic in isolation. Figure 8.2 shows an example using relative errors. The data in the figure are the result of a NIST study involving semiconductor electron mobility. The response variable is a measure of electron mobility, and the predictor variable is the natural log of the density. The fit is a rational function with a pole. Thus, since an engineer usually requires strict bounds on the maximum error it seems better to minimize the maximum error instead of the average (note that ARE $\leqslant$ MRE).

However, in the relative case, using a maximum aggregation function has its own counter-intuitive properties. For example, figure 8.3 illustrates how the zero function has a lower MRE than a model which overshoots the data, but else seems like a reasonable fit[1]. This property is particularly problematic if the model parameter space is searched automatically (hyperparameter optimization). In this case the optimization algorithm is easily deceived into generating flat models.

One would be tempted to resort to using the Maximum Absolute Error (MAE) instead. Since while it may be difficult to give a priori average error targets, giving maximum absolute error bounds is often easier since it can be related more directly to the application. However, the MAE is not a satisfactory solution either. First of all,

[1]In this case the samples are noisy but the same phenomenon can occur with noise free data and a validation set

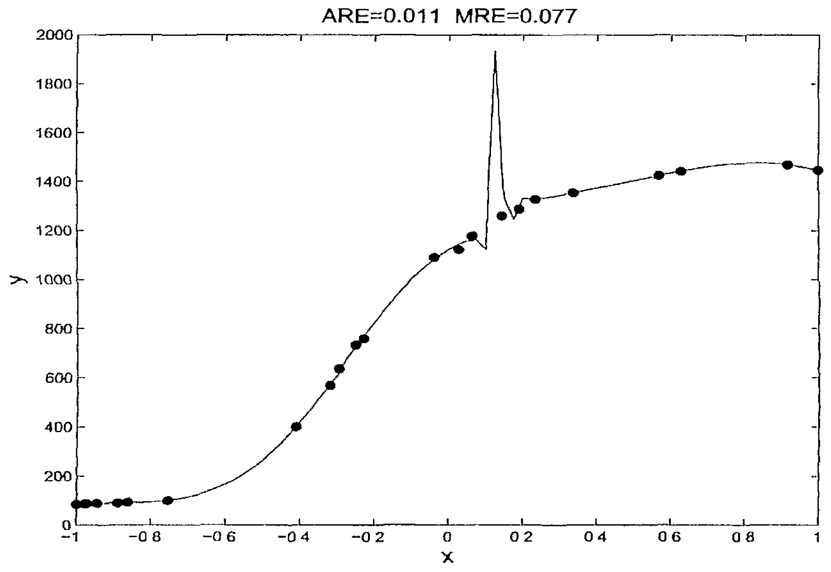


Figure 8.2: Comparison of the Maximum Relative Error and the Average Relative Error

like any absolute error, it requires knowledge of the full range of the response. Also, it is not relative, meaning a deviation of 5 on a response value 1000 is considered worse than a deviation of 3 on a value of 0.5. Furthermore, enforcing a MAE is equivalent to restricting all fitted response values $\tilde{\mathbf{y}}$ to lie inside the tube defined by $[\mathbf{y} - MAE, \mathbf{y} + MAE]$. This requirement can be too strict if the response contains regions that are very hard to fit (e.g., discontinuities), information that is not always available.

Another approach then, is to use the RRSE function, related to the popular R^2 criterion. In this case the deviation from the mean is used as the reference value.

$$RRSE(\mathbf{y}, \tilde{\mathbf{y}}) = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{8.8}$$

The RRSE is intuitively attractive since it measures how much better a fit is over the most simple model possible: the mean. Also it does not become problematic for small absolute values of y_i. Unfortunately, the problem with the RRSE is that it gives a pessimistic estimate of the error if the response that needs to be fitted is very smooth (i.e., the mean is already quite a good fit). Thus an understanding of the structure of the response is needed to properly interpret the RRSE value. The RRSE is also less intuitive for an engineer since it measures the improvement over the average model rather than the quality of fit directly (making a good choice of τ harder).

An improved function that is less sensitive to large errors and has some other attractive properties is given in [383], the Bayesian Estimation Error Quotient (BEEQ):

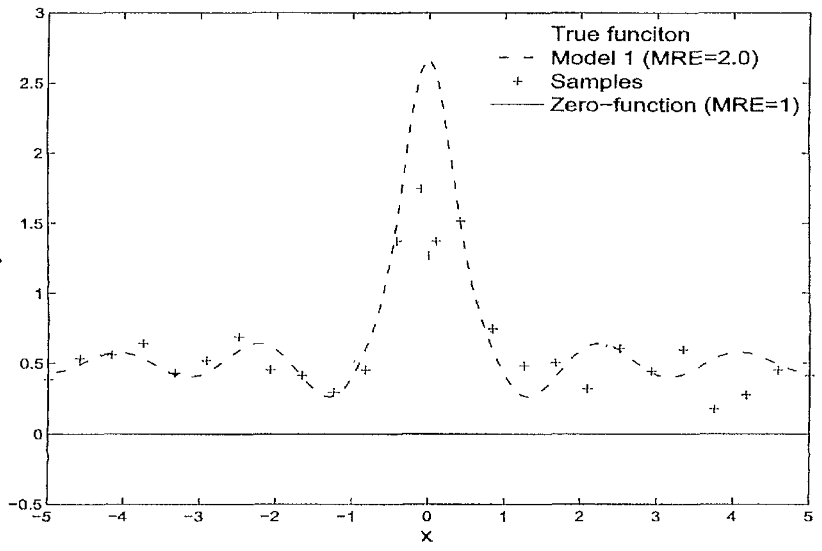


Figure 8.3: Comparison of the Maximum Relative Error over different models

$$BEEQ(\mathbf{y},\tilde{\mathbf{y}}) = \left(\prod_{i=1}^{n} \frac{\sum_{i=1}^{n} |y_i - \tilde{y}_i|}{\sum_{i=1}^{n} |y_i - \bar{y}|}\right)^{\frac{1}{n}} \tag{8.9}$$

However, like the GAE it will predict an error of zero overall if just a single point has an error of 0.

One could continue discussing different error functions (e.g., those based on the medain or mode) but it should be clear now that each error function has its own characteristics and that relative errors are not always as context free as one might assume at first. While the examples given here are quite simple, they are illustrative of the greater complexities that arise when combining an error function with a model selection metric. Also note that these subtleties are less a problem in classification (where most research on model selection is conducted). The concept of a good classifier is typically much more intuitive to grasp and define by a domain expert than in the case of regression.

Remark that the error function also influences choice of sampling strategy. For example if the error measure dictates that it is important that the optima of the model are captured accurately, one should make sure the sampling strategy employed will sample at those locations. Actually it turns out that in most cases a sampling algorithm can be formulated as a model selection critera and vica versa.

8.2.2 Choice of model selection metric

Assuming the choice of error function (and target value) can be decided upon there is still the problem of selecting a measure for estimating the generalization capabilities of a model (cross validation, bootstrap, validation set, jack-knife, etc.). This is the well known problem of model selection and has been discussed at length elsewhere [85, 171, 180, 190, 385]. A good high level introduction is given in [386]. The point this chapter attempts to make is that it is far from obvious which method to select that, when minimized, produces a model that an engineer is satisfied with. Simply using the in-sample error is useless since it does not account for over-fitting and is meaningless when used with interpolating methods (e.g., Kriging). Measures like AIC and its variants (BIC, NIC, ...) and the methods from statistical learning theory (Vapnik-Chervonenkis (VC) Dimension, etc.) are more advanced in that they take the complexity of the model into account and have solid foundations in information theory. Unfortunately, an AIC value can only be interpreted relative to another value and has no meaning on its own. They also mean very little to a domain expert.

A validation (hold-out) set is a better solution but it means there is less data available for training. Also, a hold-out error can can deceive the hyperparameter optimization if chosen poorly or if only a few points are available. For example, figure 8.4 gives a simple example where two models acheive a perfect validation score, yet only one is correct. This is of course an extreme example but similar problems are encountered with real data, particularly as the dimensionality and data sparsity increases.

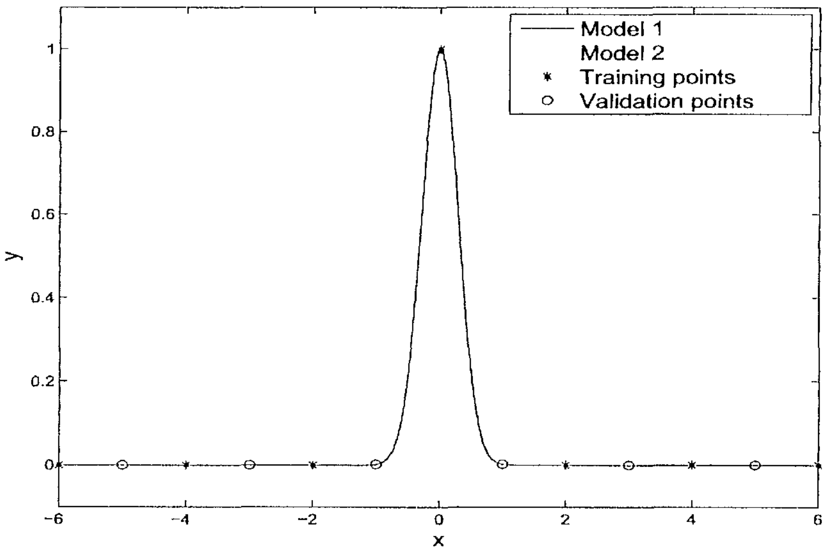


Figure 8.4: A validation set that is unable to distinguish between two models, thus potentially misleading the hyperparameter optimization.

The cross validation error (and its extreme version, the Leave-One-Out error), is a popular compromise, but it too depends on the data distribution [385, 387], can give misleading results [9], and is expensive to compute (the bootstrap even more so). Also there is the question on how to select the folds (randomly, evenly spread, etc.). Additionally one could argue the different cross validation variants should be interpreted as measuring sensitivity to loss of information rather than approximation accuracy. Finally there is the added complication of noise in the data and/or in the generalization estimator (e.g., k-fold cross validation). Since we only consider deterministic computer experiments noisy data is usually not an issue[2]. However, when dealing with measured data or stochastic simulations this adds an extra layer of complexity.

Yet a different approach is to employ Bayesian statistics (see the work by O'Hagan et. al. [125]). Through Bayesian inference one can exactly quantify the uncertainty or confidence one has in a particular model. This is usually very useful from an application standpoint but is only possible with specific model types.

The only true, unbiased test for model quality would be to assess the model on a very dense, independent test set or analytical solution. However, for any real problem this is not a feasible option since data is too expensive and an analytical solution is not available.

8.2.3 The need for handling multiple criteria

In sum, as it should be clear now, it is hard to agree upfront on a single requirement that the final replacement metamodel must respect. The fundamental reason is that an approximation task inherently involves multiple, conflicting, criteria. [383] summarizes this particularly succinctly:

> *It is an illusion that performance evaluation can be done completely fairly and impartially. This is partly because simple metrics cannot capture a complete picture of the performance of an estimation algorithm and those that are more complete [...] are more complex and subject to subjective interpretations. Also, use of any metric in performance evaluation implicitly favors the estimator that tries to optimize this same metric.*

A similar observation is made in [5]:

> *The goodness of a metamodel may not be dictated by the same performance measure but could depend on several measures. For instance, when there are many stakeholders involved in the design process, it is common that these are interested in different performance measures.*

Thus what usually happens in practice is the following: (1) a best effort is made to identify a suitable model selection metric, error function and targets; (2) simulations are performed, the model is generated and delivered to the engineer together with some statistical test results (e.g., different error metrics); and (3) the engineer visually inspects and explores the model and decides if it is satisfactory. If not the process must be repeated.

[2] In some cases discretization and convergence noise may be present, the magnitude depending on the application.

While the final evaluation stage by a domain expert should always be performed, it would be advantageous if the different desired criteria could be enforced from the start. This can be done in three main ways:

1. the different criteria (objectives) are combined into a single, global criterion which is then used to drive the model generation
2. the different objectives are enforced sequentially in a multi-level process
3. the different objectives are enforced simultaneously through a multiobjective approach

The first option is the easiest and allows existing algorithms to be re-used as is. An example of such scalarization is the geometric mean approach used by Goel et al. in [87]. However the problem remains of choosing an appropriate combination function (and its interpretation) and requiring an understanding of the ranges and nuances of the different member functions. Thus the problem is simply moved to a higher level.

The second option is a sequential or milestone approach. Multiple criteria are supported by specifying different hierarchical levels $L_1,...,L_l$ that must be reached in succession. For example, first the hyperparameter optimization process must produce a model that satisfies L_1 (e.g., a ARE of 5%). Once this target is reached, and only then, is the following level L_2 checked (e.g., a MRE of 10%). Thus, by sequentially working towards subsequent milestones, multiple criteria can be incorporated. The potential problem of this approach is that dependencies and tradeoffs between criteria can cause a deadlock (e.g., reaching one level means violating another). A different way to interpret this is as a constrained optimization problem in the hyperparameter space, each level adds a constraint. Care must be taken that there is at least one feasible region. Also the task for the optimizer (over the model parameters) becomes considerably more difficult since the optimization landscape may change suddenly and drastically when a change in level takes place.

The third solution is to tackle the problem directly as a dynamic multiobjective optimization problem in the hyperparameter space (recall that due to the incremental sampling the optimization surface is dynamic and not static). Each criterion becomes an objective and standard ranking methods are used to identify the Pareto-optimal set. The disadvantage here is that there is no longer the luxury of having a single, unambiguous best solution. However, since we noted above that such a linear ranking makes no sense this should come as no surprise. The advantage is that the problem can be tackled directly using standard algorithms. From the final Pareto set the user is then able to extract knowledge about the problem and make a better decision when choosing the final solution. In addition, the final Pareto front enables the generation of diverse ensembles, where the ensemble members consist of the (partial) Pareto-optimal set (see references in [87, 358, 388]). In this way all the information in the front can be used. An added advantage of using ensembles is that it allows the calculation of the prediction uncertainty which is very useful for an application.

Finally, one may imagine different hybrid combinations of the three methods mentioned above. For example, the multiobjective approach where the number of objectives varies dynamically. For example, when only little data is available it makes no sense to enforce application specific criteria, or force the model response into particular bounds. That makes more sense when sufficient data is available and the model uncertainty has been reduced. Other combinations are possible but this is a topic that has seen little research and that goes beyond the scope of this chapter. Rather we shall concentrate on the multiobjective approach.

8.3 Modeling multiple outputs

The previous section described how a multiobjective approach to global surrogate modeling can help solve *the 5 percent problem*. A second use case is when dealing with multi-output systems. It is not uncommon that a simulation engine has multiple outputs that all need to be modeled [8, 129]. For example, the combustion problem described in [389] has both a chemical and temperature source term that needs to be modeled. Also many Finite Element packages generate multiple performance values simultaneously.

The direct approach is to model each output independently with separate models (possibly sharing the same data). This, however, leaves no room for trade-offs nor gives any information about the correlation between different outputs[3]. Instead of performing two modeling runs (doing a separate hyperparameter optimization for each output) both outputs can be modeled simultaneously if models with multiple outputs are used in conjunction with a multiobjective optimization routine. The resulting Pareto front then gives information about the accuracy trade-off between the outputs in hyperparameter space and allows the practitioner to choose the model most suited to the particular context. More arguments, of essentially the same discussion, are given in [391].

Again, multi-output Pareto based modeling enables the generation of diverse ensembles. This is a popular approach in rainfall runoff modeling and model calibration in hydrology [67, 392]. Models are generated for different flow components and/or derivative measures and these are then combined into a weighted ensemble or fuzzy committee. A Pareto based approach to multi-output modeling also allows integration with the automatic surrogate model type selection algorithm described in chapter 7. This enables automatic selection of the best model type (Artificial Neural Network (ANN), Kriging, Support Vector Machine (SVM), ...) for each output without having to resort to multiple runs [370, 393]. This is illustrated in figure 8.5.

[3]For example, it is well-known that in linear regression analysis (multivariate) Generalized Least Squares (GLS) dominates (in a Pareto front sense) Ordinary Least Squares (OLS) in linear regression analysis [8]. The downside that GLS is more complicated is mitigated by Rao's theorem [390] that shows that GLS reduces to OLS computed per output if the same design matrix is used.

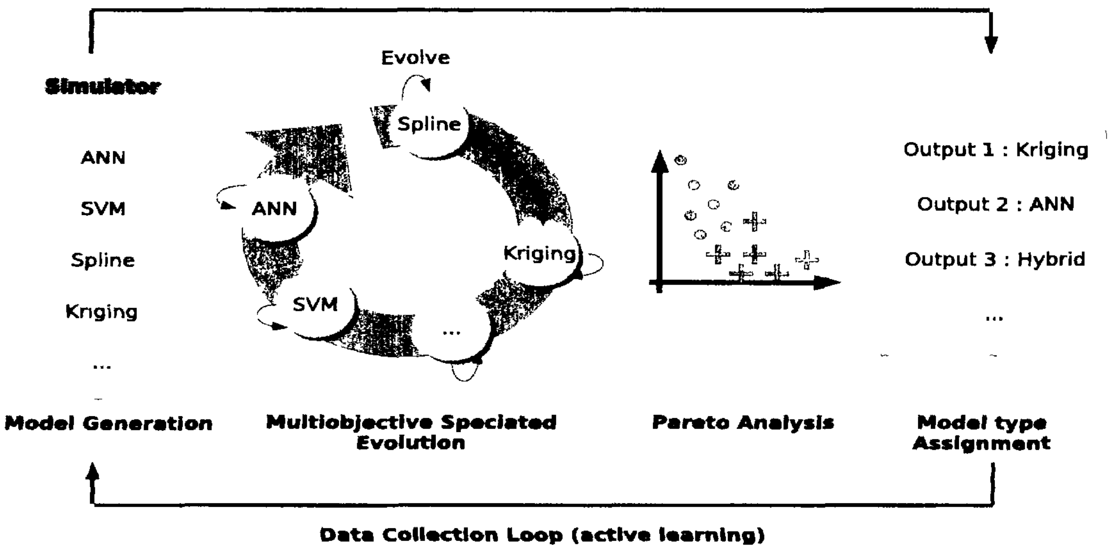


Figure 8.5: Multi-objective model generation with automatic model type selection for each output.

8.4 Related work

There is a vast body of research available on single objective hyperparameter optimization strategies and model selection criteria for different model types: [180, 182, 183, 191, 345–347] and the extensive work by Yao et. al. [350, 351]. Many authors have noticed the problems with single objective hyperparameter optimization but it is only very recently that multiobjective versions of classical machine learning methods have been presented [394–397]. An extensive and excellent overview of the work in this area is given by Jin et. al. in [388] and the book (edited by Jin) [398]. By far the majority of the cited work uses multiobjective techniques to improve the training of learning methods. Typically an accuracy criterion (such as the validation error) is used together with some regularization parameter or model complexity measure (e.g., the number of support vectors in SVMs) in order to produce more parsimonious models [384]. Other criteria used include: sensitivity, specificity, interpretability, and number of input features [388, 395].

It seems less work has been done on high level objectives (with error functions and generalization estimators in particular) that do not depend on a particular machine learning method. [399] optimize an accuracy metric (the RMSE) together with an application specific *Return* metric useful for stock market forecasting. An example of the use of multiple error measures (and incidentally one of the first formulations of multiobjective learning) is [400] who minimized the L_2-norm, the L_∞-norm and a complexity measure. Unfortunately, a single-objective GA was employed to perform the optimization, resulting in only a single solution. [401] also give an example with two error functions, the Euclidean and robust error which they use to fit a noisy sinusoid with an ANN.

Few references are available that explicitly deal with the trade-offs between different error functions for surrogate modeling. [384] agree that determining the error function is key but do not consider the problem any further. [402] give an extensive treatment of 15 popular error functions for time series extrapolation but is of little use for regression. A more relevant and extensive overview is given by Li and Zhao in [383] who discuss many practical metrics for performance estimation in general and propose a number of new ones. A more restricted and philosophical discussion is given in [403]. The previous references are mainly theoretical. Empirical results on performance estimation are harder to find. One example is [404] who compare four different error functions used for neural network classification training.

Another topic that has been the subject of extensive research is that of multiobjective surrogate based optimization (MOSBO). Surrogate methods are widely used for the optimization of expensive functions [405]. While initially their use has been constrained to the single objective case, an increasing number of results are being reported in the multiobjective case. An example is the work on statistical improvement by Keane et. al. [260] and ParEGO [259], the multiobjective version of the popular Efficient Global Optimization (EGO) approach [89]. Another example is the application to parameter optimization of earth system models in [406], for crashworthiness design optimization in [407], for robust optimization in [98], and for thin wall structure optimization in [408]. The well known NSGA-II algorithm [409] has also been extended to incorporate surrogate models [261,410]. In this context some work has also been done on comparing different performance measures for use in MOSBO [352,411]. [352] compare different performance criteria for improving metamodel based optimization. They also *"...recognize that in order to obtain desirable information or knowledge about a response surface, multiple performance measures taken in concert may be necessary."* Unfortunately they stop there and do not discuss the issue any further. Though the research into MOSBO is still young, an excellent overview of current research is already available in [262].

In this chapter we stress the importance of a critical analysis of performance estimation criteria and the associated trade-offs when generating surrogates (optimizing the hyperparameters). In particular, a founded choice of error function and target is often overlooked and performance estimation is done in a more ad hoc manner [22,24], constrained to a single objective [19,23,25], or done a posteriori (after the model parameters have been fixed) to compare different models [19,412]. While the implications and trade-offs of different performance criteria are well described in the statistics community (e.g., [402]), the resulting insights and possible solutions can use more visibility.

8.5 Applications

This section presents some concrete illustrations of the ideas discussed in the previous sections. This section will focus on multi-output modeling and multi-output model type selection. An application of multi-criteria model selection will be discussed in chapter 9.

8.5.1 Analytic function

8.5.1.1 Background

To easily illustrate the concepts and potential problems we first take a predefined analytical function with two input parameters x_1, x_2 and two responses y_1, y_2:

$$y_1(\vec{x}) \quad = \quad (1-x_1)^2 + 100(x_2 - x_1^2)^2 \tag{8.10}$$

$$y_2(\vec{x}) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{d} \cdot \sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d} \cdot \sum_{i=1}^{d} \cos(2\pi \cdot x_i)\right) + 20 + e \tag{8.11}$$

So $f(x_1, x_2) = [y_1, y_2]$ with $x_i \in [-2, 2]$ and $d = 2$. Readers may recognize these two responses as representing the Rosenbrock and Ackley functions, two popular test functions for optimization. Plots of both functions are shown in figure 8.6. We choose

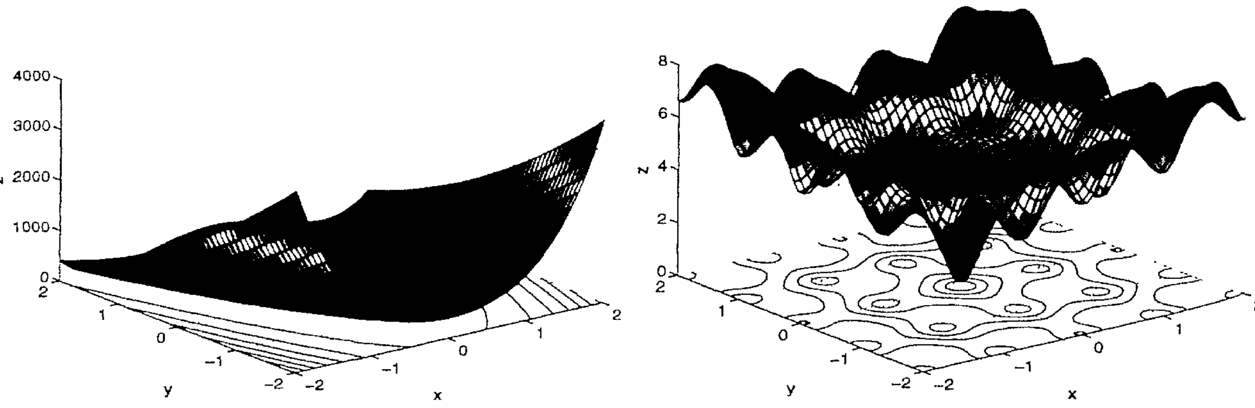


Figure 8.6: The Rosenbrock (left) and Ackley (right) functions

this combined function since it is an archetypal example of how two outputs can differ in structure. Thus it should show a clear trade-off in the hyperparameter space when modeled together (i.e., a model that accurately captures y_1 will be poor at capturing y_2 and vica versa).

It is important to stress that we are ***not*** interested in optimizing these functions directly (as is usually done), rather we are interested in reproducing them with a surrogate model (with two inputs and two outputs), using a minimal number of samples. Thus the problem is effectively a dynamic multi-objective optimization problem in the *hyperparameter* space.

8.5.1.2 Experimental Setup

We start with the multi-objective hyperparameter optimization problem as a first use case. The Kriging [270] and NSGA-II plugins were used. The correlation parameters (θ) represent models in the population. Following general practice, the correlation function was set to Gaussian, and a linear regression was used. Starting from an initial Latin Hypercube Design of 24 points, additional points are added each iteration (using a density based active learning algorithm) up to a maximum of 150. The density based algorithm was used since it is shown to work best with Kriging models [142]. The search for good Kriging models (using NSGA-II) occurs between each sampling iteration with a population size of 30. The maximum number of generations between each sampling iteration is also 30.

As second use case for this analytical problem we consider the automatic model type selection plugin. The following model types were included in the evolution: Kriging models, single layer ANNs (based on [413]), Radial Basis Function Neural Networks (RBFNN), Least Squares SVMs (LS-SVM, based on [269]), and Rational functions. Together with the ensemble models (which result from a heterogeneous crossover, e.g., a crossover between a neural network and a rational function), this makes that 6 model types will compete to fit the data. In this case, the multi-objective GA implementation of the Matlab GADS toolbox is used (which is based on NSGA-II). The population size of each model type is set to 10 and the maximum number of generations between each sampling iteration is set to 15. Again, note that the evolution resumes after each sampling iteration. In all cases model selection is done using the Root Relative Square Error (RRSE) on a dense validation set.

8.5.1.3 Results

Two snapshots of the Pareto front at different number of samples are shown in figure 8.7. The full Pareto trace (for all number of samples) is shown in figure 8.8a. The figures clearly show that the Pareto front changes as the number of samples increase. Thus the multi-objective optimization of the hyperparameters is a dynamic problem, and the Pareto front will change depending on the data. This change can be towards a stricter trade-off (i.e., a less well defined 'elbow' in the front) or towards an easier trade-off (a more defined 'elbow'). What happens will depend on the model type.

From the figure it is also immediately clear that the Rosenbrock output is much easier to approximate than the Ackley output. Strangely though, there seems to be

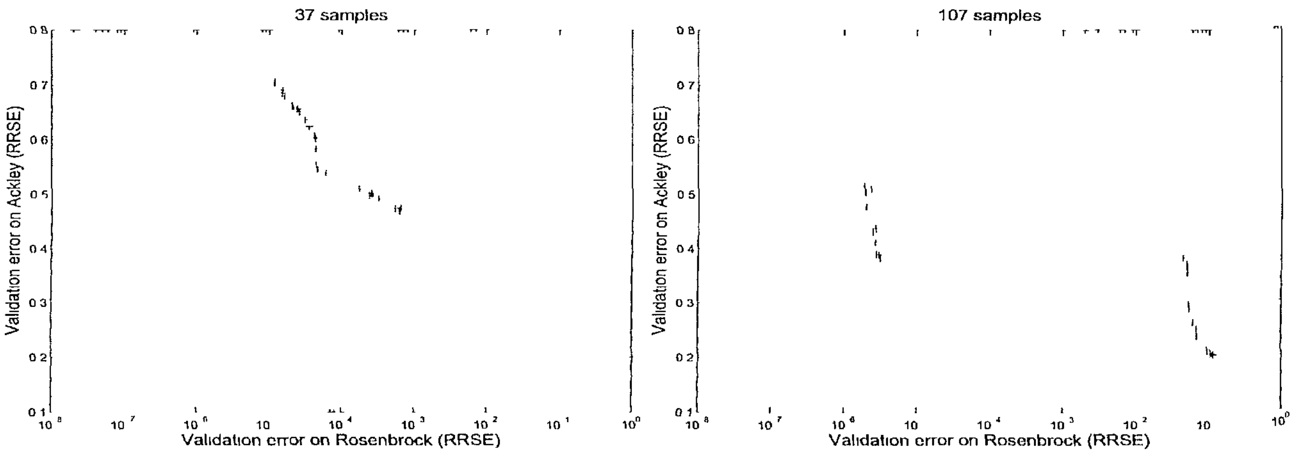
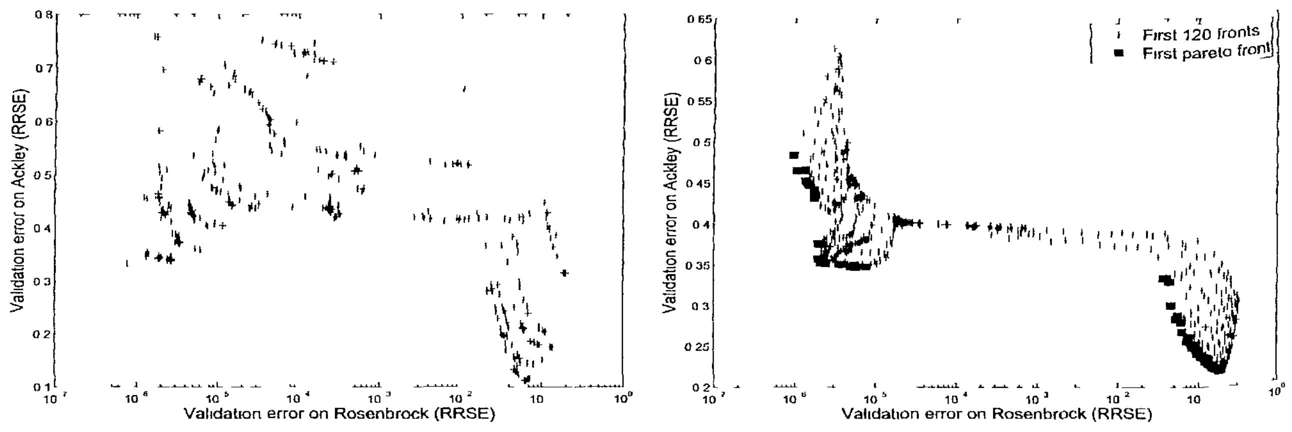


Figure 8.7. Two snapshots of the Pareto front (during the model parameter optimization) at different sampling iterations (AF)



(a) With sample selection (up to 150 samples) (b) Without sampling (brute force hyperparameter search at 124 samples)

Figure 8.8: Pareto front search traces (AF)

a discontinuity in the front. The Pareto front is split into two parts and as sampling proceeds the algorithm (NSGA-II) oscillates between extending the left front over the right front (or vica versa). The full Pareto trace in figure 8.8a also shows this.

To understand what is causing this behavior, a brute force search of the hyperparameter space was performed for a fixed LHD of 124 sample points. The space of all possible θ parameters was searched on a 100x100 grid with bounds [-4 3] (in log_{10} space) in each dimension. Simultaneously an extensive NSGA-II Kriging run was performed on the same data set for 450 generations. In both cases a dense validation set was used to calculate the accuracy of each model. The combination of both searches (for both outputs) is shown in figure 8.9 (note that the RRSE is in log scale). The brute force search of the θ-surface also allows the calculation of the true Pareto front

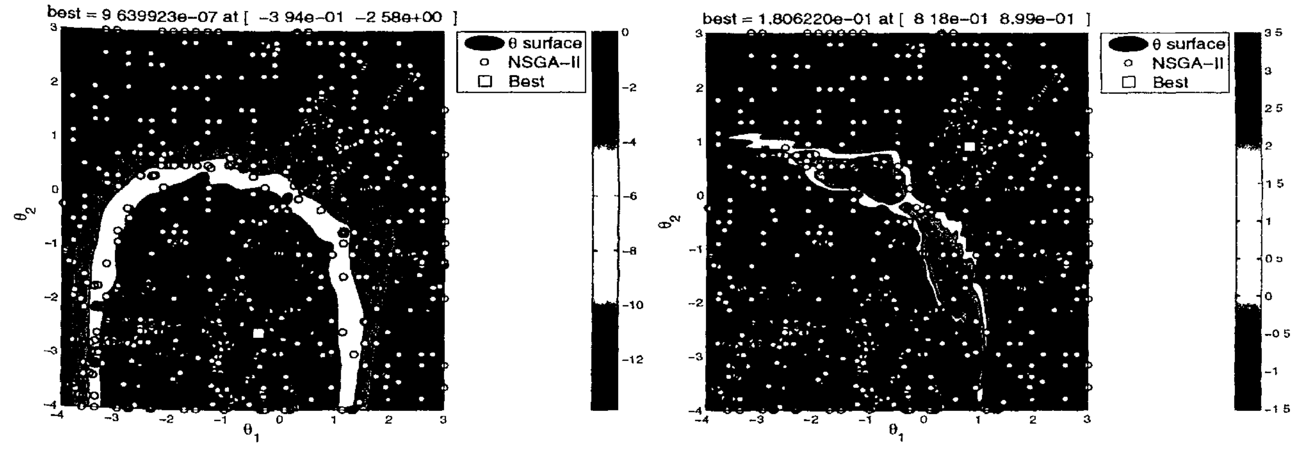


Figure 8.9: Kriging θ-surface with the NSGA-II search trace (AF, left: *Rosenbrock,* right: *Ackley)*

(by performing a non-dominated sorting). The resulting Pareto front (and the next 119 fronts, these are shown for clarity) are shown in figure 8.8b.

Studying the surfaces in figure 8.9 reveals what one would expect intuitively: the Rosenbrock output is very smooth and easy to fit, so given sufficient data a large range of θ values will produce an accurate fit. Fitting the Ackley output, on the other hand, requires a much more specific choice of θ to obtain a good fit. In addition both basins of attraction do not overlap, leading to two distinct optima. This means that (confirming intuition) the θ-value that produces a good model for y_1 produces a poor model for y_2 and vice versa. Together with figure 8.8b this explains the separate Pareto fronts seen in figure 8.7. The high ridge in the Ackley surface makes that there are no compromise solutions on the first front. Any model whose performance on y_2 would lie between the two separate fronts would never perform well enough on y_1 to justify a place on the first front. Thus, the fact that NSGA-II does not find a single united front is due to the structure of the optimization surface and not due to a limitation of NSGA-II itself.

The analytic problem was also tackled with the automatic model type selection algorithm described in [267]. This should enable the automatic identification of the most adequate model type for each output *without* having to perform separate runs. Figure 8.10 shows the full Pareto search trace for the test function across all sampling iterations.

The figure shows the same general structure as figure 8.8a: there is a strong trade-off between both outputs resulting in a gap in the search trace. If we regard the model selection results we find they agree with intuition. The Rosenbrock function is very easily fit with rational functions, and its smooth structure makes for an excellent approximation with almost any degree assignment (illustrated by the straight line at roughly 10^{-7} on the x-axis). However, those same rational models are unable to produce a satisfactory fit on the Ackley function, which is far more sensitive to the

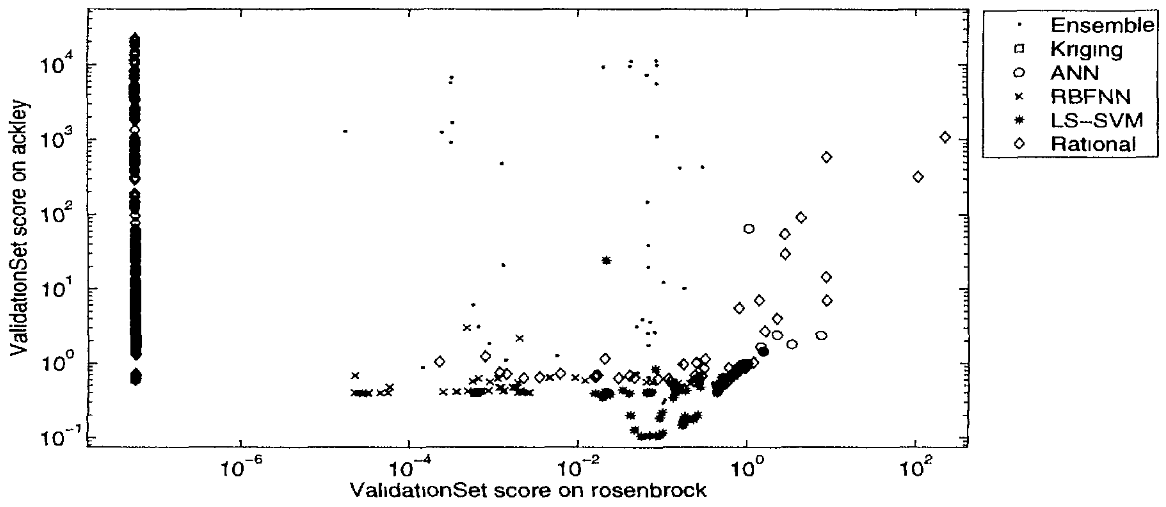


Figure 8.10: Heterogeneous Pareto trace (AF)

choice of hyperparameters. Instead the LS-SVM models perform best, the RBF kernel function matching up nicely with the different 'bumps' of the Ackley function.

Thus, we find the results to agree with intuition. The Rosenbrock function is better approximated with a global model since there are no local non-linearities. While the Ackley function benefits more from a local model, and this is exactly what figure 8.10 shows. Note the diverse front shown in the figure now allows the generation of a diverse ensemble (further improving accuracy) using standard ensemble techniques (bagging, weighted, etc.).

8.5.2 Low Noise Amplifier

8.5.2.1 Background

As a second application we revisit the LNA modeling problem from chapter 5. Recall that the performance figures of a LNA (e.g. voltage gain, linearity, noise figure, etc.) can be determined by means of computer simulations and are functions of the design parameters (e.g. width and length of transistors, bias conditions, values of passive components) [268]. The goal of the design process is to figure out one or more sets of design parameters resulting in a circuit which fulfills the specifications, i.e. constraints given on the performances. For this example we now consider the modeling of the performance figures directly. We consider the two-dimensional case where the power consumption P and third-order linearity $IIP3$ are approximated in function of the transistor width W and the inductance Lm.

8.5.2.2 Experimental setup

The same settings were used for this problem as for the analytical function, except that the sample selection loop was switched off and LS-SVM models were used to fit the data (instead of Kriging). Instead of sampling, a 12^2 full factorial design was used.

8.5.2.3 Results

The analytic function is of course just a synthetic example, but it is useful as an illustration and for pointing out potential problems. When we now turn to the LNA problem we shall see that the same kind of problems can arise when modeling real data. Figure 8.12a shows the search trace of an LS-SVM hyperparameter optimization run using NSGA-II (RBF kernel, optimizing c and σ). Of course the trade-off is not as extreme as with the analytic example but we do note some similarities: one output (P) is significantly easier to model than the other and there is again a gap in the Pareto front. A brute force search of the LS-SVM hyperparameter space (with the NSGA-II results overlaid) confirms this (figure 8.11). It is also interesting to compare figure 8.11 with figure 8.9. Both figures are in line with the authors' experience that optimizing the Kriging θ parameters is typically harder than selecting good values for c, σ. Our experience is that the θ optimization landscape is more rugged, multi-modal, and more sensitive to the data distribution (see also the discussion in [142]). On the other hand, it is usually quite easy to generate an LS-SVM that captures the trends in the data without being too sensitive to the data distribution. An added benefit of SVM type models is that the number of hyperparameters is independent of the problem dimensionality (unlike Kriging).

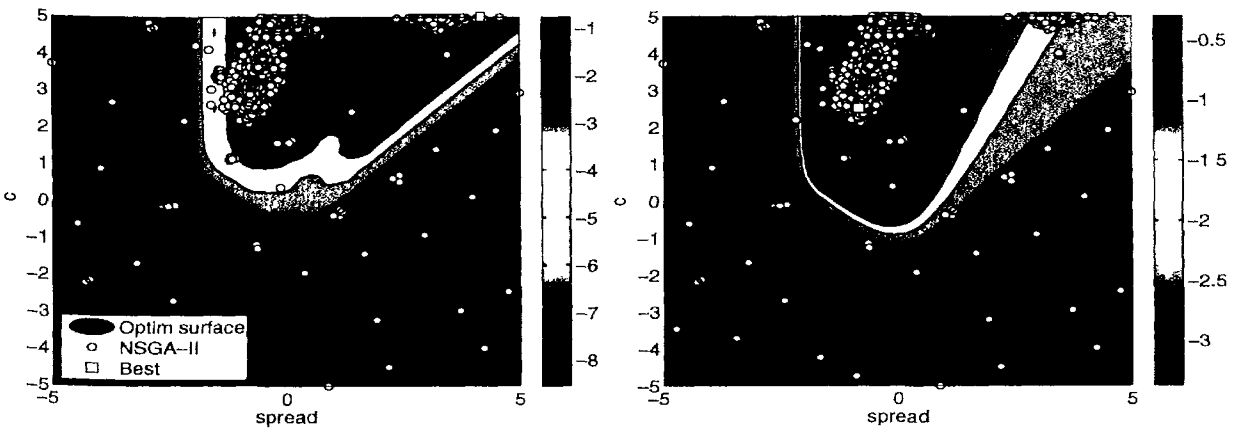
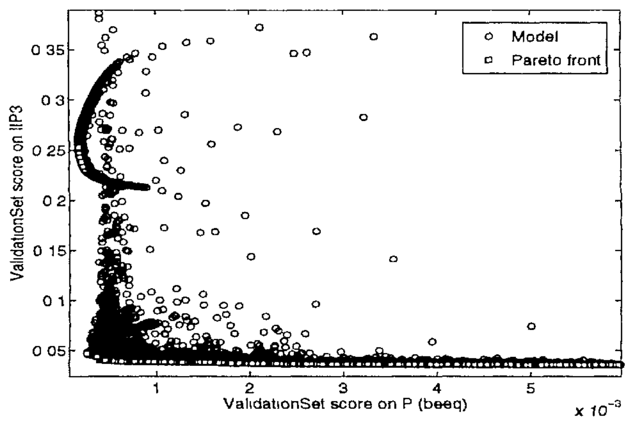
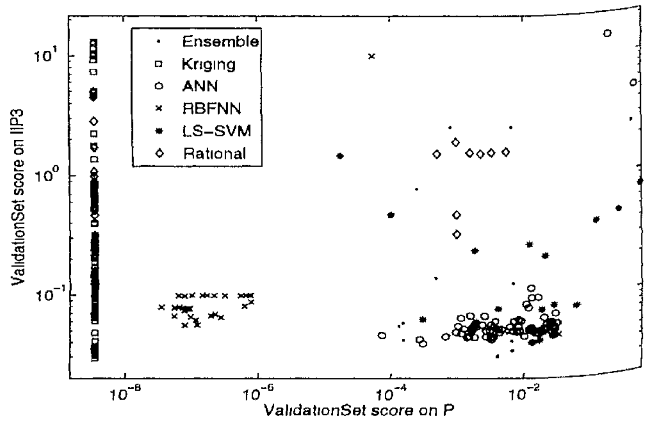


Figure 8.11: SVM (c, σ)-surface with the NSGA-II search trace (LNA, left: *P,* right: *IIP3)*

Finally, we can also apply the automatic model type selection algorithm to the LNA data. The resulting search trace is shown in figure 8.12b. Again, note the similarity with figure 8.10. However, contrary to the previous problem, we see that in this case



(a) SVM($\$c, \sigma$)-optimization search trace and Pareto front (LNA)



(b) Heterogeneous Pareto trace (LNA)

Figure 8.12: Pareto front search traces

including different model types can actually alleviate the trade-off in the data. It turns out that while LS-SVM models have problems capturing both outputs accurately, the Kriging models are able to do so, producing a single best model for both outputs. Again the almost straight line at 10^{-8} indicates that the P output is extremely easy to fit, a wide range of hyperparameters will produce a good fit.

8.5.3 Automotive problem

The third example is an application from the automotive industry (see [25] for details).

8.5.3.1 Background

Today the early concept phase of a car body development process is marked by the optimal coordination of design specifications with the requirements on the mechanical behavior of the structure as well as on the feasibility. This planning process is repetitive for the same body parts and the solution finding is carried out mostly by experience with an additional virtual tryout afterwards in order to improve the solution. The use of surrogate modeling can enable an early feasibility prediction of body parts.

The geometry of a B-pillar bottom of a side frame is shown in figure 8.13. There you have a recurring feasibility challenge in sheet metal forming that can be explained by radii, depths and angles as experience shows. Which of these geometry parameters and in which combination they have an effect on the feasibility is, however, intuitively hard to predict. For simulation purposes the door entry area can be separated from the side frame by simple boundary conditions without major restrictions for the validity of the analysis but computing times considerably go down.

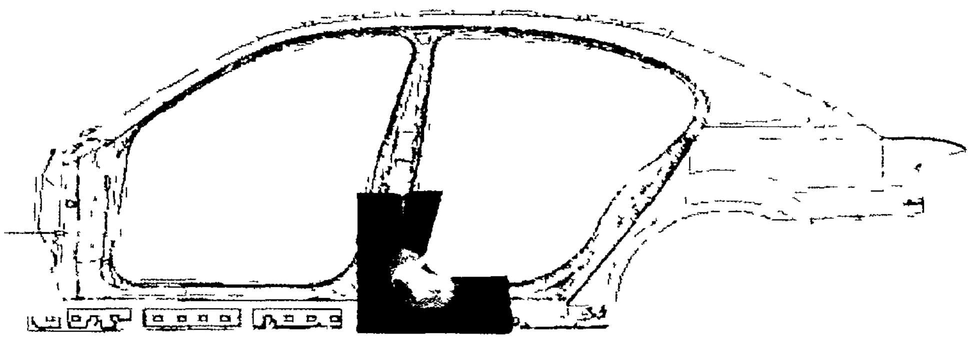


Figure 8.13: B-pillar bottom of a side frame [25]

The entry angle α_1, opening angle α_2, frame depth h and entry radius r, have been chosen as geometry parameters (see [25] for details). In addition, for every geometry constellation the blank boundary was determined so that the forming result was optimal. Additional process parameters, like draw bead or blank holder forces, have not been used. So there were six parameters that have been taken into account. With these input quantities a sampling based on a LHC was created. Maximum scaled distances of the strain states to the forming limit curve and a maximum thinning limit respectively were chosen as output quantities indicating feasibility. The data sampling phase resulted in 1998 data points evaluated that were suitable for modeling. The overall target for this particular problem setting was to predict a given set of geometry parameters as feasible, i.e., to predict the existence of cracks (*cracking* output) or unacceptable thinning (*thinning* output).

8.5.3.2 Experimental setup

Both outputs shall be modeled together using the ANN and LS-SVM plugins of the SUMO Toolbox. The ANN models are based on the Matlab Neural Network Toolbox and are trained with Levenberg Marquard backpropagation with Bayesian regularization [271,272] (300 epochs). The topology and initial weights are determined by a GA. The LS-SVM models are based on the LS-SVMlab toolbox plugin [269] and the hyperparameters are searched in log_{10} space with $\sigma \in [-4,4]$, $c \in [-5,5]$ (an RBF kernel is used). The multiobjective algorithm used is the one implemented in the Matlab GADS toolbox which, in turn, is based on NSGA-II. The population size is set to 10. For comparison each output will be modeled separately as well (single objective).

In all cases the metric used to drive the hyperparameter optimization is the Average Relative Error (ARE) on 5-fold cross validation. For the single objective runs the timeout was 25 generations, for the multiobjective runs the timeout was 50 generations.

8.5.3.3 Results

Figure 8.14 shows the final error curves after the SUMO Toolbox has terminated. A point is plotted for each time the toolbox finds a model that improves on the previous model. As can be seen from the figure, the ANN models clearly outperform the SVM models, especially for the *cracking* output. One could argue the poor performance of the LS-SVM models is due to poor hyperparameter optimization. However, this is not the case. For reference a brute force search of the hyperparameter landscape was conducted on a 50 by 60 grid. This is shown in figure 8.15 (bounds in log_{10} scale, the white crosses show the area explored by the SUMO Toolbox). The minimum found through this search:

$$f_{cracking}(-0.1600, -1.4993) = 0.1348$$

$$f_{thinning}(0, -2.9996) = 0.0730$$

is comparable to the minimum found by the SUMO Toolbox:

$$f_{cracking}(-0.2173, 0.2978) = 0.1280$$

$$f_{thinning}(0.0423, 1.0948) = 0.0741$$

Thus the hyperparameter optimization is not to blame (remember that the cross validation procedure introduces some noise into the surface). A more extensive cross validation (15 folds) was also done on the final best model in each case (table 8.1). As can be seen, the accuracy remains unchanged.

	cracking	*thinning*
ANN	0.0414	0.0325
LS-SVM	0.1305	0.0741

Table 8.1: ARE on 15-fold cross validation of the final models (automotive example)

The poor performance of SVMs in this case is in line with the author's previous experience. We found SVM models to require too much data when a non-linear, noise-free response needs to be fitted smoothly and accurately. In those cases, SVM models are very good at fitting the non-linear regions but generate unwanted 'ripples' in the regions where the response needs to be smooth or data is sparse. ANN models on the other hand, are able to adapt much better to the heterogeneity of the response. The sigmoid transfer functions allow for high non-linearity, while proper training (e.g., through the use of regularization) ensures a smooth fit in the sparse regions.

Figure 8.16 shows the full trace of the multiobjective hyperparameter optimization. In this case the model generation is driven by a 2-objective (the cross validation score

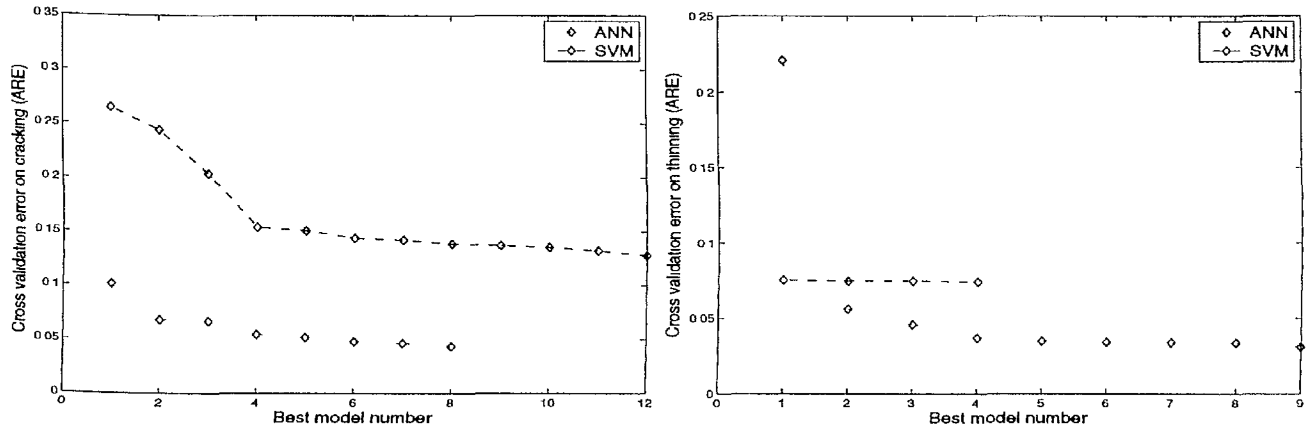


Figure 8.14: Model accuracies in the single objective case (left: cracking, right: thinning)

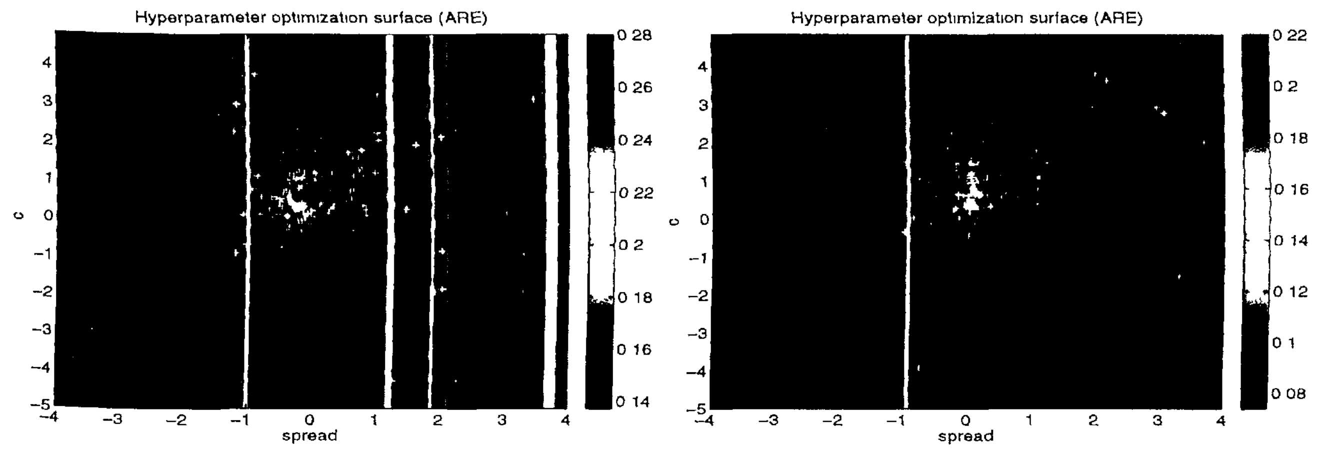
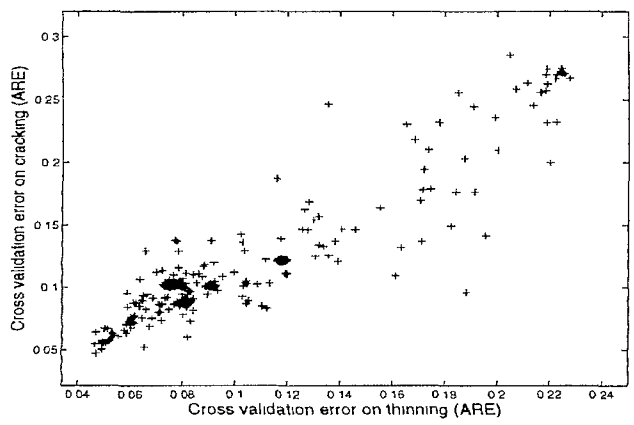


Figure 8.15: SVM hyperparameter optimization surface with the NSGA search trace (left: cracking, right: thinning)

on each output) optimization algorithm. In both cases it is immediately clear that there is no real Pareto front, a single best model can be identified in each case. Thus this teaches us that there is a very strong correlation between both outputs and that good performance on one output, implies good performance on the other. This is actually to be expected since *cracking* and *thinning* are closely related (as can also be seen from figure 8.15).

Of course this is not always the case (see for example [393]). It is not always clear how much the outputs are really correlated, or how much one quality metric influences another (in the case of multiple metrics). We argue that in those cases a direct multiobjective approach should be considered. It is guaranteed to give at least as much information as doing multiple single objective runs for about the same computational cost (which is still outweighed by the cost of the simulation). Also, it gives the engineer more flexibility and is a cleaner approach than manually combining the multiple



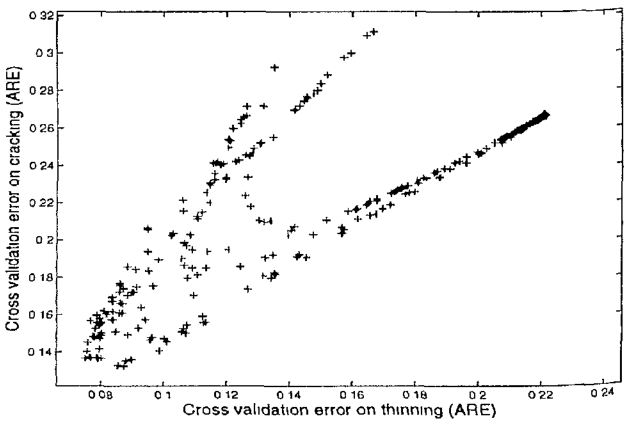


Figure 8.16: Model accuracies in the multiobjective case (left: ANN, right: SVM)

objectives into a single formula.

8.5.4 Langley Glide-Back Booster

8.5.4.1 Backgrond

The last application is a modeling problem from aerodynamics (see also section 7.8.4). We use it to focus on the automatic model type selection per output. NASA's Langley Research Center is developing a small launch vehicle (SLV) [376] that can be used for rapid deployment of small payloads to low earth orbit at significantly lower launch costs, improved reliability and maintainability. In particular, NASA is interested in the aerodynamic characteristics (lift, drag, pitch, side-force, yaw, roll) as a function of three inputs (Mach number, angle of attack, and side slip angle). Simulations are performed with a Cart3D flow solver with a running time of 5-20 hours on a high end workstation. A fixed data set of 780 adaptively chosen points was generated for metamodeling purposes. Thus the active learning loop is switched off.

8.5.4.2 Experimental setup

For the modeling the following model types are used: Artificial Neural Networks (ANN), rational functions, Kriging models [270], and RBF LS-SVMs [269]. The result of a heterogeneous recombination will be an ensemble. So in total 5 model types will be competing to approximate the data.

The ANN models are based on the Matlab Neural Network Toolbox and are trained with Levenberg Marquard backpropagation with Bayesian regularization (300 epochs). The topology and initial weights are determined by the Genetic Algorithm (GA). For the LS-SVMs the c and σ are selected by the GA as are the correlation parameters for Kriging (with a linear regression and Gaussian correlation function). The rational functions are based on a custom implementation, the free parameters being the orders of the two polynomials, the weights of each parameter, and which parameters occur in the denominator.

The population size of each deme type is set to 15 and the GA is run for 50 generations. The migration interval m_i is set to 5, the migration fraction m_f to 0.1 and the migration direction is *forward* (copies of the m_f best individuals from island i replace the worst individuals in island $i+1$). The fitness function is defined as the Bayesian Estimation Error Quotient calculated on a 20% validation set where the minimal distance between validation points is maximized.

8.5.4.3 Results

For ease of visualization all outputs were modeled in pairs. Figure 8.17 shows the resulting search trace for three representative combinations (the others were omitted to to save space).

Let us first regard the $lift - drag$ trace. We notice a number of things. First all the models are roughly on the main $x = y$ diagonal. This means that a model that performs well on $lift$ performs equally well on $drag$. This implies a very strong correlation between the behavior of both outputs, which can actually be expected given the physical relationship between aerodynamic lift and drag. Only the rational models do not consistently follow this trend but this has probably more to do with their implementation as will be discussed later. Since all models are on the main diagonal the model type selection problem has become a single objective problem. It turns out that the ANN models (or actually an ensemble of ANN models if you look closely) are the best to use in this case, performing much better than the other model types. Kriging and SVM perform about the same with SVM doing slightly better.

The situation in the second plot ($lift - yaw$) is different. There is a clear trade-off between both outputs. The ANN models still perform best on $lift$ but the yaw output is fitted most accurately by rational models and their ensembles. The difference between SVM models and Kriging models is now more marked, with the SVM models performing much better on the yaw output. Remark also that the distribution of the rational models is much more spread-out and chaotic than that of the other model types (which are relatively clustered). This could actually be seen for all output combinations. The reason is due to the way the order selection of the rational functions is implemented. The order selection contains too much randomness and not enough exploitation of the search space, leading to erratic results.

The $pitch - roll$ plot has the same general structure, though the $pitch$ output turns out much harder to approximate than $lift$. It is interesting to see though that ensembles turn out to be significantly superior to standalone models. It turns out that most of these ensembles turn out to be combinations of the best performing model types.

The results presented so far are of course static results. In reality the best performing model type changes as the evolution progresses. This can be seen from figure 8.18

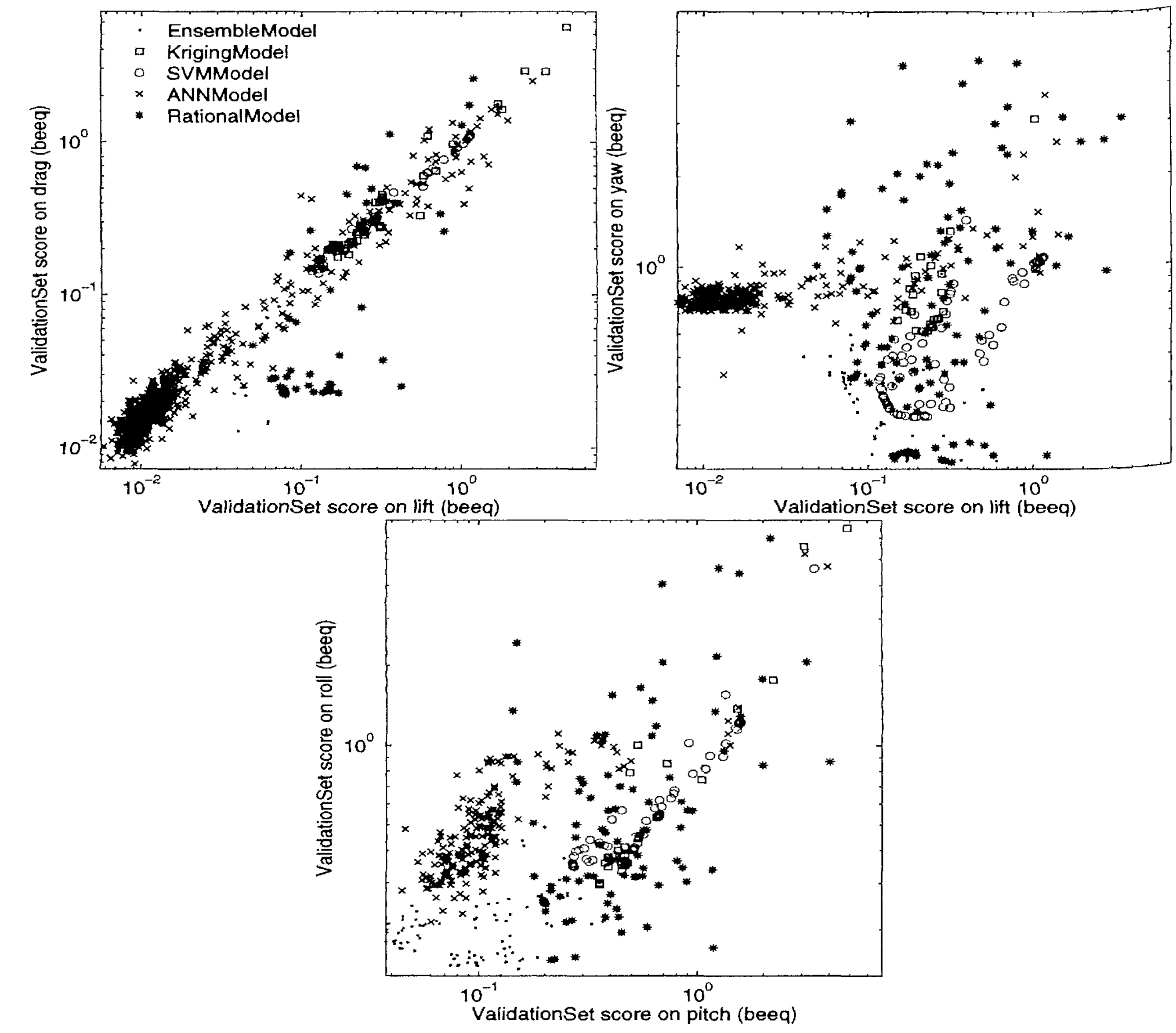


Figure 8.17: Heterogeneous Pareto traces: lift — drag (left), lift — yaw (right), pitch — roll (bottom)

which shows the relative share of each model type of the total population for two cases. Model types that do well will have more offspring, thus increase their share. This dynamics is particularly interesting and useful in the context of active learning. It means the optimal model type changes with time, depending on how much information (data points) are available. This is exactly what one would expect (see [267] and references therein for details and examples in the single objective case).

In sum the algorithm seems to have done quite well in detecting the correlations between the outputs, and the model selection results agree with what one would expect intuitively from knowledge about the structure of the responses. The obtained accura-

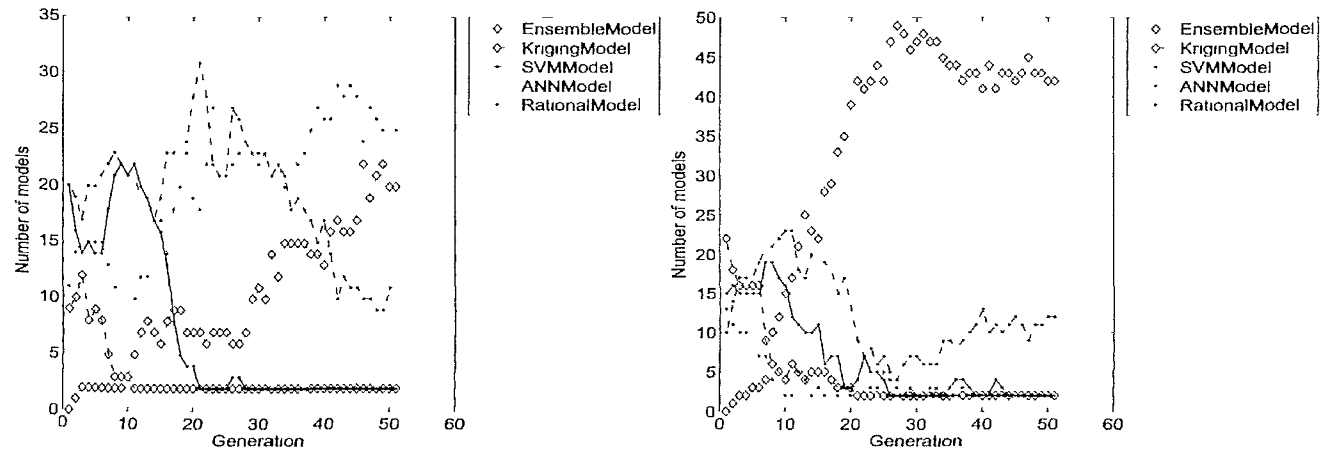


Figure 8.18: Evolution of the population make-up lift —yaw (top) and pitch —yaw (bottom)

cies are also similar to those obtained by single model type runs. It is interesting to see that, while the performance of Kriging and SVM was very similar (which is to be expected given their connection to Gaussian Process (GP) theory), the SVM results were consistently better and more robust to changes in their hyperparameters. The exact reasons for this are being investigated. It was also quite surprising to see the ensemble models to so well. In virtually all experiments the final Pareto front consisted of only of ensembles. Depending on the model selection process they are predominantly homogeneous (containing multiple models of the same type) or heterogeneous (thus 'filling in' gaps in the Pareto front).

8.6 Summary and conclusion

A crucial problem of generating global surrogate models for a particular application (or any function approximation task for that matter), is agreeing upfront with the domain expert what criteria the final surrogate should satisfy. The problem is that each criterion (encompassing an error function, generalization estimator, and target value) involves a tradeoff between interpretability, accuracy, bias, and computational efficiency. Thus, for cases where this trade-off cannot be inferred from domain knowledge or application constraints a multiobjective approach to solving this problem should be considered. The advantage of a multiobjective approach is also that it allows multiple outputs to be modeled together, giving information about the tradeoff in the hyperparameter space. It further enables the generation of diverse ensembles and the application of an automatic model type selection algorithm. This enables each output to be automatically modeled with the most suitable model type. There is also some empirical evidence that the number of local optima can be reduced by converting multi-modal single-objective problems, into multiobjective ones [388]. If the same can be proven in machine learning it means the task of identifying good surrogate models can become easier through a multiobjective approach.

However, a disadvantage of the multiobjective approach is that as the number of dimensions (criteria/outputs) increases the solution space increases exponentially [414]. Thus the search for the Pareto optimal set becomes harder, requires more search iterations, and the final set is more cumbersome for the practitioner to explore and understand. For costly simulation codes the extra computational effort is negligible, and good GUI tools can help a domain expert understand the relationships present in the Pareto-optimal set. However, for cheaper codes a trade off between simulation cost and modeling cost will have to be made. The poor scalability of non-dominated sorting algorithms above 4 dimensions is also an issue [262]. Luckily, advances in algorithms [415,416], *front vizualization* [417], *and gains in computational efficiency* [418] continue to be made.

A disadvantage of the multiobjective approach versus the milestone approach is that the direct multiobjective approach takes all criteria into account straight away. This is not necessarily a problem but is not always the most computationally efficient. For example, in the case of adaptive sampling it makes no sense to check or enforce an (expensive) application specific constraint if only a few data points are available. The model first needs to mature by incorporating more data before undergoing more stringent checks. In this case the number of objectives varies dynamically and thus a scalarized multiobjective approach with a dynamically varying weighting parameter (as discussed in [419]) can be useful. Alternatively a cooling approach as done in [157] could be used.

Thus, naturally much work remains. First of all, while support for multiple criteria is already very useful, more research is needed on intuitive criteria. Ideally criteria should be easily formulated in language that a domain expert is comfortable with and fully understands. Fuzzy theory can be helpful in this respect. Besides researching the feasibility of fuzzy criteria more work still needs to be done on classic model selection methods and explore the relationship with a constraint based approach. This to fully understand the relationship between error function and generalization estimator, and how they impact the final response. A way to vary the criteria dynamically with the sample selection loop would also be useful as is the study of transductive learning [420]. A possible integration with domain partitioning methods (e.g., as done in [198]) is also promising.

Another area requiring further investigation is understanding how the iterative sample selection process influences the hyperparameter optimization landscape. There is a mutual dependency between the model type, hyperparameter optimization strategy, and sampling strategy (e.g., see [142]). The exact nature of this dependency depends on the model type. Determining how they interact and may be optimally combined is a topic of ongoing research. For the tests in this chapter the authors have simply let the optimization continue from the previous generation. However, some initial tests have shown that an intelligent restart strategy can improve results. Knowledge of how the number and distribution of data points affects the hyperparameter surface (determined by some metric) would allow for a better tracking of the optimum, reducing the computational cost. The influence of noise and discrete variables on the hyperparameter optimization (e.g., neural network topology selection) also remains an issue.

In general, while some progress towards dynamic multiobjective optimization has been made [421, 422], this is a topic that current research in multiobjective surrogate modeling is only just coming to terms with [262]. Or as English pithily puts it: *"Optimization is easy, learning is hard (in the typical function)."* [423].

9 Estimating Response Nonlinearity in Regression

Entia non sunt multiplicanda praeter necessitatem.

(Entities must not be multiplied beyond necessity)

– William of Ockham

9.1 Introduction

A recurring theme in this dissertation has been that the primary users of global surrogate modeling methods are domain experts, whose main concern is to obtain an accurate replacement metamodel for their problem as fast as possible and with minimal overhead. Thus if surrogate modeling methods are to see widespread, industrial adoption by non-modeling specialists it is important for the model complexity selection problem be tackled in an autonomous way. As discussed in chapters 3 and 8, this makes the choice of the model quality estimation metric crucial.

A popular choice in this respect are resampling strategies such a cross validation [170]. While cross validation almost always performs very well [424] it is expensive to use and is not always efficient at preventing unwanted ripples or bumps in the final model response. These lead the domain expert to mistrust or even reject the model and hamper its re-use in the engineering pipeline. This chapter presents a new measure, called Linear Reference Model or LRM, that helps reduce this problem. By resampling the model it provides a non-parametric estimate of the observable (as opposed to structural) nonlinearity of model. When used alone or as an auxiliary penalty during the hyperparameter optimization process it can lead to more accurate and parsimonious models over classic methods like cross validation. It is also computationally cheap to evaluate and is independent of the model type.

9.2 Background and Motivation

9.2.1 Background

Many model selection criteria have been developed and discussed at length in the literature: re-sampling methods (cross validation, bootstrap, leave-one-out, hold-out set, etc.) [170, 424, 425], information theoretic methods (AIC, MDL, BIC, etc.) [171], and methods from statistical learning theory [178, 191, 426]. Much work has also been done on hyperparameter optimization strategies for different model types, both in the single objective case [180, 182, 183, 191, 345–347, 351] and multi-objective case [394–397, 427]. An excellent overview of this line of work is given by Jin et. al. in [388]. In the multi-objective case, typically an accuracy criterion (such as the validation error) is used together with some regularization parameter or model complexity measure (e.g., the number of support vectors in SVMs) in order to produce more parsimonious models [384]. Other criteria used include: sensitivity, specificity, interpretability, and number of input features [388, 395]. An alternative approach is to use model types and metrics fully rooted in Bayesian statistics, as is done in the work by Kenneth, and O'Hagan [125]. In this case the model generation procedure does not result in a single best model, but rather a distribution over all possible models obtained through Bayesian inference.

In general, though, there is no universally optimal model selection procedure and much depends on the model type and on the amount and distribution of data points [9]. Thus surrogate modeling researchers typically employ re-sampling methods such as (leave-one-out) cross validation [170] since they are generic, easy to apply, and generally quite accurate [424]. It has been shown, though, that the generalization error obtained through re-sampling methods can be quite biased depending on the data distribution and re-sampling strategy [9, 425, 428].

9.2.2 Structural versus Observable Complexity

Chapter 5 described a benchmarking study that assessed how well different model types were able to approximate a parametrized RF circuit block under a fixed adaptive sampling strategy and increasing dimensionality. The study used an unbiased test set as model selection metric which, in a next step, needed to be replaced by some approximation (in any real problem such a test set is not available). However, tests showed that, while resampling methods could be used to estimate the model accuracy, they are expensive and the final models sometimes show unwanted ripples or bumps in their response.

An example of unwanted behavior is the ridge shown in figure 9.1. Naturally these artefacts become more problematic as the dimensionality and data sparsity increases.

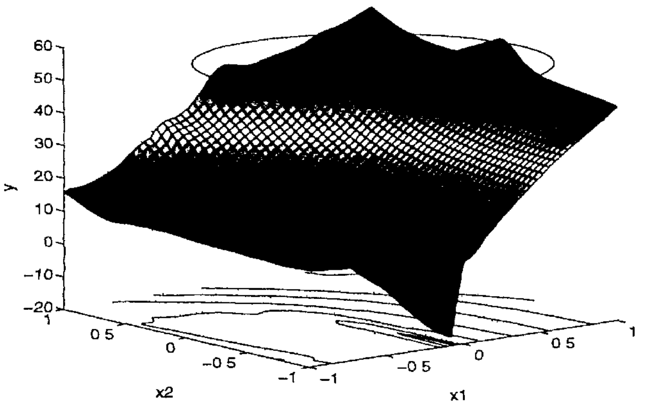
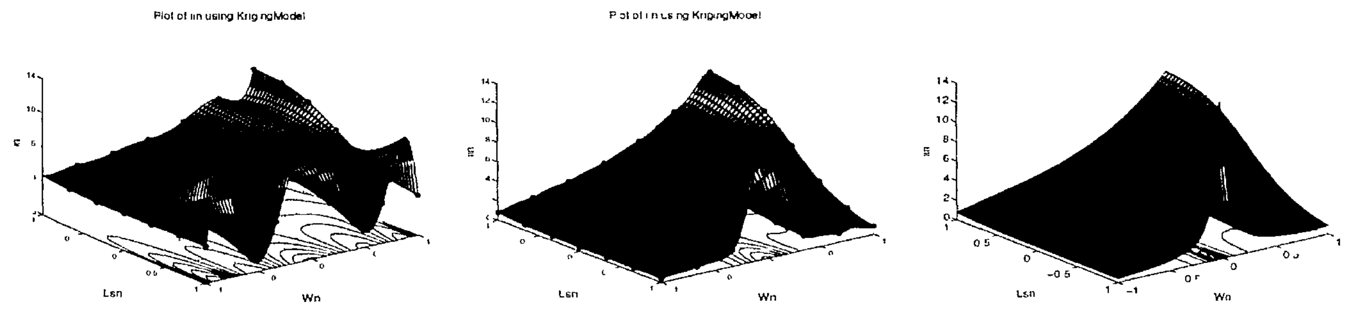


Figure 9.1: 2D slice of a 6D ANN model for a problem from electronics. The ridge at the top of the figure is an example of an undesirable artifact that we want to avoid.

Another example is shown in figure 9.2a. The figure results from generating a Kriging model that minimizes the root mean square error on a 20% max-min validation set (the minimum distance between the validation points is maximal). The true function is shown in figure 9.2c. Figure 9.2a clearly shows large oscillations where the true behavior should be much flatter and smoother. Figure 9.2b shows the result if we minimize the LRM metric proposed in this chapter. The full Pareto search trace showing

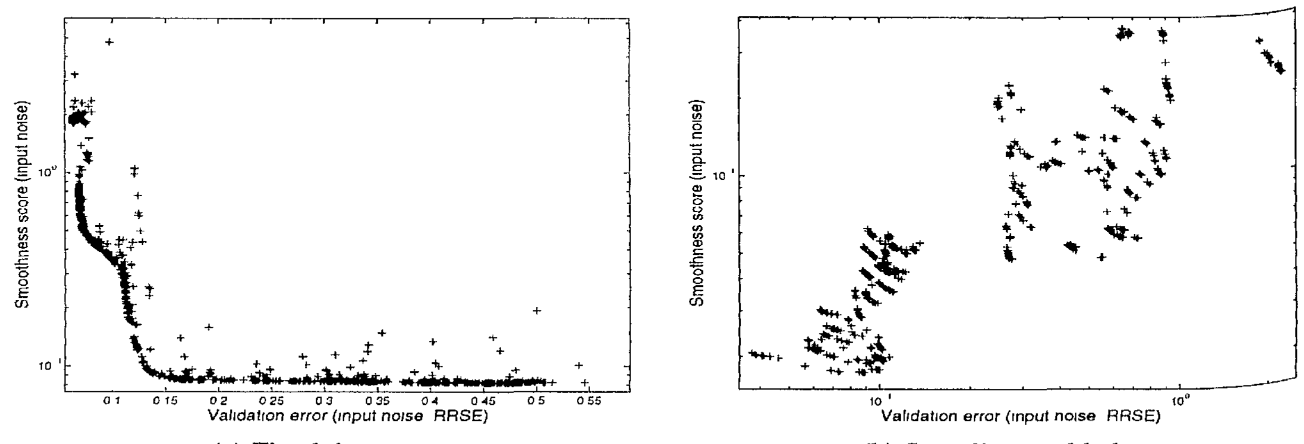


(a) Minimum max-min validation error (20% of the training data) (b) Minimum LRM score (c) True function

Figure 9.2: Surrogate models of the input noise current ($\sqrt{i_{in}^2}$) of a Low Noise Amplifier [20] generated with different model selection criteria. The dots represent the training data.

the trade-off for a fixed data set is shown in figure 9.3a.

Recall from chapter 8 that the Pareto front will change if sampling is enabled.



(a) Fixed data set (b) Sampling enabled

Figure 9.3: Pareto search trace for the LNA problem

This is shown in figure 9.3b[1]. The figure clearly shows how the front advances and the model quality improves as more data becomes available. In addition the trade-off in the front seems to decrease as the number of points increase. This should be expected since as the amount of data increases there is less uncertainty about the correct hyperparameter values and the agreement between both measures increases.

9.2.3 Computational cost

A second disadvantage of re-sampling strategies is that they are computationally expensive to evaluate, for some model types (e.g., large neural networks) the resulting modeling cost may even outweigh the simulation cost. On the other hand, they are attractive since they are very easy to implement, generally give good results, and do not depend on the model type. For example, this makes them useful when dealing with domain specific approximation techniques or when dealing with proprietary modeling code where the model implementation or structure is hidden.

9.2.4 Motivation

These findings drove us to search for a metric that (1) tackled the problem of unwanted oscillations or ridges in models directly, (2) was computationally cheap(er) to evaluate, and (3) was independent of the approximation method. The goal is not to replace existing criteria but rather to augment their performance where necessary.

In effect, the surrogate model shown in figure 9.2a exhibits high behavioral complexity, i.e., the behavior that we can *observe* looks complicated. This complicated

[1]A movie showing the evolution is available at `http://sumolab.blogspot.com/`

behavior can be, but does *not* necessarily have to be, due to a high structural complexity of the model, i.e., the model formulation itself is complex. It is important to note that the implication is not absolute (e.g., the *sinc* function). Naturally there is a certain correlation between the complexity of the response and the complexity of the representation (e.g., the number of model parameters) but it depends on the domain of interest and the model type (e.g., the correlation for polynomial functions is much stronger than for neural networks).

In the Genetic Programming (GP) community these two types of complexity are known as phenotype complexity and genotype complexity [342, 429]. The majority of work on model selection has concentrated on structural, genotype complexity. Many metrics have been designed that explicitly take into account the complexity of the model structure or formulation, penalizing models with a large number of parameters or terms. Probably most well known are information theoretic measures like MDL, AIC, and its variants [171]. In GP many metrics have been defined that quantify the complexity of the tree or graph that represents a solution in the population [342,430,431]. These are then typically combined with some goodness of fit criteria in a single or multi-objective framework [429]. Remark, though, that intuitive assumptions that the structural complexity should always be minimized and that a model with a large number of parameters relative to the available data is by definition undesirable do not always hold [432–434].

Much work has also been done in the statistical learning theory community on data and class dependent model complexity metrics. Data-dependent complexity measures have been developed as an alternative to function-class-dependent complexity measures (such as the fat-shattering dimension). Examples include discrepancy-based measures, Rademacher complexity measures, and Gaussian complexity measures [435, 436]. Usually these measures are used to build bounds driving model selection, and may intervene as regularization terms in a structural risk minimization approach. An overview of these and other regularization related topics is given in [425,437].

Less work has been done on measuring the phenotype complexity directly. Work on estimating function smoothness has been done in [438], based on techniques from signal analysis (i.e., looking at the frequency content of a function) [425]. Recently, [439] propose a new model selection metric based on the modulus of continuity of a function [440] and provide upper bounds for the modulus of continuity for different estimation functions. From GP, a new complexity metric is introduced in [342] that measures the order of non-linearity of a given GP tree. It is based on the degree of the best fitting polynomial of the symbolic equation represented by the tree. Thus it is a kind of hybrid between genotype and phenotype complexity. However, due to numerical constraints the algorithm does not scale well beyond two dimensions and is thus applied on each sub-tree independently (univariate case only).

9.2.5 The LRM Assumption

In global surrogate modeling the goal is to capture the true behavior of the simulation code as efficiently and accurately as possible. Any visible behavioral complexity (bumps, ripples, etc.) should be explainable by data points at (or in the vicinity of) those locations. Thus, intuitively, if there is no such evidence nearby, the bump should be regarded as an artifact of the model. We propose a new behavioral complexity metric that is based on the following core assumption:

If nothing else is known, the model behavior between two neighboring points should be linear.

We term this Linear Reference Model (LRM) based model selection. In a nutshell, the LRM metric penalizes a model proportional to how much it deviates from a linear fit. At first this assumption may seem misguided, a perfect linear fit is usually not desirable as a surrogate model since it is not continuous and needs far too much data to result in a usable approximation. However, typically it is very difficult for a non-linear surrogate model (Kriging, SVM, ANN, RBF model, etc.) to produce a linear interpolation. The final model will always be continuous (unless the fitting model itself contains discontinuities of course). Thus the LRM assumption should be interpreted as a kind of penalty or parsimony pressure [431] that will push the hyperparameter optimization into the direction of a linear fit but will typically not reach the goal itself.

The idea of calculating the deviation from a linear fit in order to identify nonlinear regions of the response is also briefly described in [158]. There it is used in a weighting scheme in order to perform efficient data reduction. Its formulation as a model or sample selection criterion is, unfortunately, not exploited.

As a final note, it is important to remark that while the LRM assumption can be useful for global surrogate modeling it may be less useful for surrogate driven optimization since the LRM assumption can hamper the models' prediction of new optima[2]. In this case a link with the prediction uncertainty is desirable.

9.3 LRM Metric for Response Nonlinearity Estimation

We now introduce the LRM algorithm in detail. But first the authors stress that priority should always be given to model specific complexity or regularization measures if available. For surrogate models where a model specific approach is infeasible or insufficient, the proposed measure can be used to improve results.

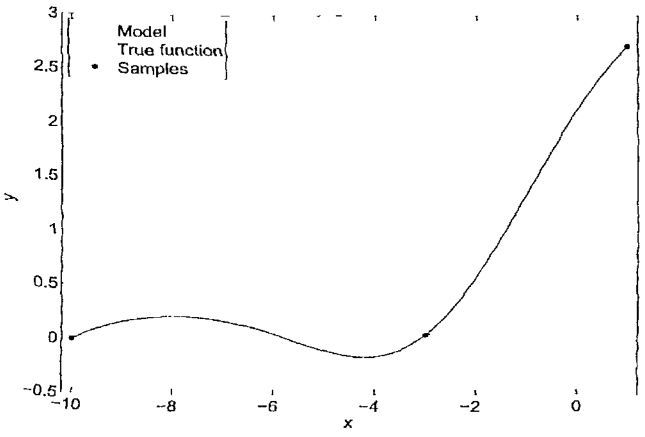
[2]Replacing the linear interpolation by a quadratic one would work but this would require an extra data point (or domain knowledge) to figure out if the interpolation should 'point up' or 'point down'.

9.3.1 Concept and illustration in 1D

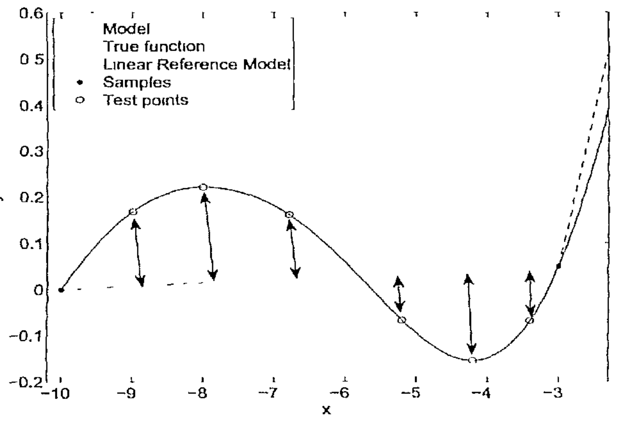
To illustrate the LRM concept we take a simple predefined mathematical function with only one input parameter x and one output y:

$$y = f(x) = e^x, \tag{9.1}$$

with $x \in [-10, 1]$. A plot of this function can be see in figure 9.4a.



(a) Mathematical function and approximation



(b) Deviation of the approximation from a linear interpolation based on a number of test points (enlarged for clarity).

Figure 9.4: Intuitive 1D illustration of the LRM metric

Consider a model approximating the function using 3 samples (at $x = -10, -3$, and 1). Regarding the prediction in figure 9.4a, the model shows some unwanted oscillations at the left part of the function. The core assumption underlying LRM is then to penalize the model for that behavior. To determine how much the model should be penalized the linearity of the model can simply be quantified as the perpendicular distance between the model and the linear interpolation[3]. To that end, the linear interpolation is evaluated at several test points for each set of sample points. The magnitude of the distances (denoted by the arrows in figure 9.4b) are then aggregated and returned as the final penalty for that model[4].

If the underlying simulator also produces derivative information together with sample data (remember we do not want to run extra simulations just for model selection) this helps but does not void the need for a linear interpolation. One still needs to see what happens in between points. In addition, if the surrogate model type supports the calculation of the prediction uncertainty this can also be used to help infer to what degree the oscillation is justified.

[3]Note that using a quadratic interpolation is not possible since it would require additional simulation points or re-training the model

[4]Readers may note the link with sampling strategies, this point will be revisited in section 9.5.

9.3.2 Generalization to d dimensions

Denote the set of samples used to construct the surrogate model $\tilde{f}(x)$ as a matrix X,

$$X = (\mathbf{x}_1, \ldots, \mathbf{x}_k)^T = \begin{pmatrix} x_{1,1} & \ldots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{k,1} & \ldots & x_{k,d} \end{pmatrix} \tag{9.2}$$

where each row is exactly one sample point. The core LRM assumption is that if nothing else is known we assume linear behavior in between two neighboring sample points (keeping in mind the caveats from section 9.2). The more a model deviates from this, the more it should be penalized.

The first step is to construct a d-dimensional linear interpolation of X. This can be done by constructing a Delaunay triangulation [441] on the k samples used for constructing the surrogate model. A Delaunay triangulation is a triangulation such that no sample is inside the circum-hypersphere of any simplex. In addition it has the property that it is unique if no $d+1$ samples are on the same hyperplane and no $d+2$ samples are on the same hypersphere.

Let s_i be a simplex ($i = 1..l \leq k^{\lceil \frac{d}{2} \rceil}$) defined by samples $P_i = (\mathbf{p}_1, ..., \mathbf{p}_{d+1})^T$ in barycentric coordinates. The simplex s_i, together with the corresponding output values of its corner points, is used to construct an unique interpolating hyperplane H_i with coefficients $\mathbf{a}_i = (a_{i_1}, \ldots, a_{i_{d+2}})$. Of interest is then the function $h(\mathbf{x})$ that represents the perpendicular distance between H_i and $\tilde{f}$:

$$h(\mathbf{x}) = d_{\perp}(\tilde{f}(\mathbf{x}), H_i(\mathbf{x})) \tag{9.3}$$

Note that we take the perpendicular distance and not the norm $||\tilde{f}(\mathbf{x}) - H_i(\mathbf{x})||_2$ since the norm penalizes the steep parts of the response too severely. A plot of $h(\mathbf{x})$ for the 1-dimensional example from the previous section is shown in figure 9.5. Ideally the area under $h(\mathbf{x})$ for each simplex s_i should be minimized:

$$\min \left(\int .. \int_{s_1} h(\mathbf{x})\, d\mathbf{x} + .. + \int .. \int_{s_l} h(\mathbf{x})\, d\mathbf{x} \right) \tag{9.4}$$

Since this integral can typically not be calculated exactly (the analytical expression of a surrogate model is not always easy to obtain or integrate) an approximation would be needed: one of the many quadrature rules in low dimensions, or Monte Carlo/sparse grid based methods for higher dimensions. However, this quickly becomes complicated in higher dimensions, especially since we are dealing with simplices and not orthotopes. Thus we opt to minimize the mean of $h(\mathbf{x})$ instead. This is much cheaper and has a similar effect. To estimate the mean we use a Monte Carlo approach and sample $h(\mathbf{x})$ using a space filling (in the linear interpolation space) test point distribution. We are most interested in speed and capturing the largest oscillations of the function rather than capturing the small variations accurately. Let points $\mathbf{b}_1, ..., \mathbf{b}_{d+2}$

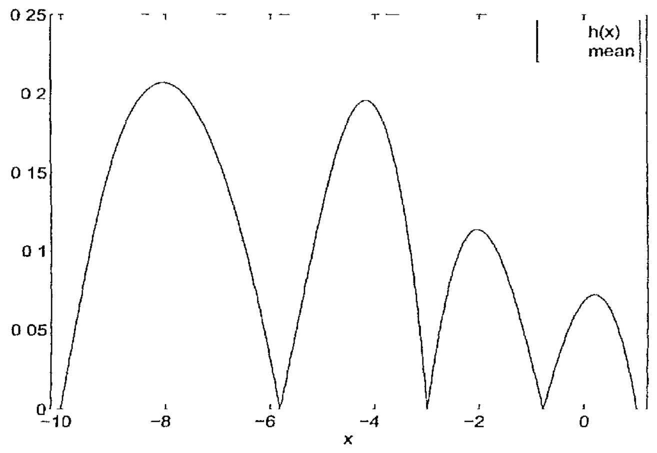


Figure 9.5: A plot of $h(\mathbf{x})$ from equation (9.3) is shown for the 1-dimensional example from figure 9.4a.

be the test points selected for each simplex s_i. The first point chosen is the geometric centroid. Using barycentric coordinates this can be calculated as

$$\mathbf{b}_1 = \frac{\sum_{j=1}^{d+1} \mathbf{p}_j}{d+1} \tag{9.5}$$

The remaining $d+1$ test points are chosen halfway on the simplex medians

$$\mathbf{b}_{j+1} = \frac{\mathbf{p}_j + \mathbf{b}_1}{2}, j = 1 \ldots d+1 \tag{9.6}$$

A graphical illustration of the resulting distribution in the 2D case is shown in figure 9.6. Note that this procedure is different from what is typically done in lazy learning [218–220]. In lazy learning test points are selected first after which the optimal neighborhood (and possibly local model type) is searched for in order to produce a prediction. Here we do the reverse. Another difference is that we are not interested in the quality of the prediction of the local linear model itself. Rather we are interested in how much the global surrogate model deviates from it (versus work such as [442]).

Subsequently, for each simplex s_i the test points $\mathbf{b}_j$ are converted to Cartesian coordinates ($j = 1, \ldots, d+2$)

$$\mathbf{c}_j = \mathbf{b}_j \cdot P_i \tag{9.7}$$

and placed into a matrix Q_i together with the response value predicted at each $\mathbf{c}_j$:

$$Q_i = \begin{pmatrix} c_{1,1} & \cdots & c_{1,d} & \tilde{f}(\mathbf{c}_1) & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ c_{d+2,1} & \cdots & c_{d+2,d} & \tilde{f}(\mathbf{c}_{d+2}) & 1 \end{pmatrix} \tag{9.8}$$

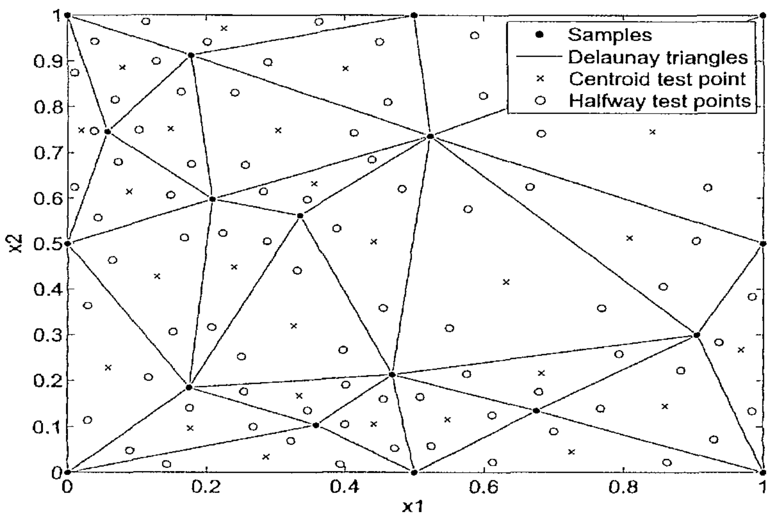


Figure 9.6: Test points in a 2D Delaunay triangulation

This allows the calculation of the penalty score for this simplex s_i. This penalty score is the perpendicular distance between $H_i(\mathbf{x})$ and $\mathbf{q}_j$, where $\mathbf{q}_j$ is the j^{th} row of Q_i.

$$score_i = \frac{1}{d+2} \sum_{j=1}^{d+2} \frac{|\mathbf{a}_i \cdot \mathbf{q}_j|}{||\mathbf{a}_i||_2} \tag{9.9}$$

Note other aggregation functions are possible over the simple average we employ here (e.g., the maximum). The scores $score_i$ for each simplex s_i are then aggregated to obtain the overall LRM score. Again, we use the mean here, but other aggregations are possible.

$$LRM(\tilde{f}(x)) = \frac{1}{l} \sum_{i=1}^{l} score_i \tag{9.10}$$

Finally, remark that under loose assumptions of continuity

$$\lim_{|X| \to \infty} LRM(\tilde{f}(\mathbf{x})) = 0 \tag{9.11}$$

Our experience is that approximation methods have most trouble with responses where the response behavior is heterogeneous (not uniformly nonlinear/'bumpy' and not uniformly linear/smooth). It is in these cases that we expect LRM to be most useful.

9.3.3 Computational Cost

The dominating cost of the algorithm is performing the triangulation in order obtain an d-dimensional linear interpolation. Fortunately, this only has to be calculated when the data distribution changes (once every sampling iteration[5]). Thus many surrogate models can be measured with no (significant) additional cost. In contrast, re-sampling methods need to be calculated for every model separately.

The implementation used in this chapter is based on QHull [443]. QHull obtains the Delaunay triangulation for a set of points by constructing the convex hull in one dimension higher than the dimension of the points. This, by adding the length of each point as the last coordinate, effectively creating a paraboloid out of the points. After creating the convex hull the lower part of the hull is projected to the original space resulting in the Delaunay triangulation.

According to the QHull documentation the average time complexity to construct a convex hull in d dimensions is $O(\frac{k.f_v}{v})$, where k is the number of points, v is the number of output vertices and $f_v$is the maximum number of facets for a convex hull of v vertices. The latter, f_v, is tightly bound to the dimension of the problem and grows rapidly as the number of dimensions increase.

Before we discuss concrete running times, keep in mind that in our experience a very rough empirical limit to the application of generic (i.e., not domain specific) global surrogate modeling methods with sequential sampling is a few thousand points in 6 dimensions. Naturally this depends on the problem and method at hand. The fundamental problem, though, is that as the dimensionality increases the increase in data points needed to maintain the same high modeling accuracy (typically two significant digits) becomes impractical.

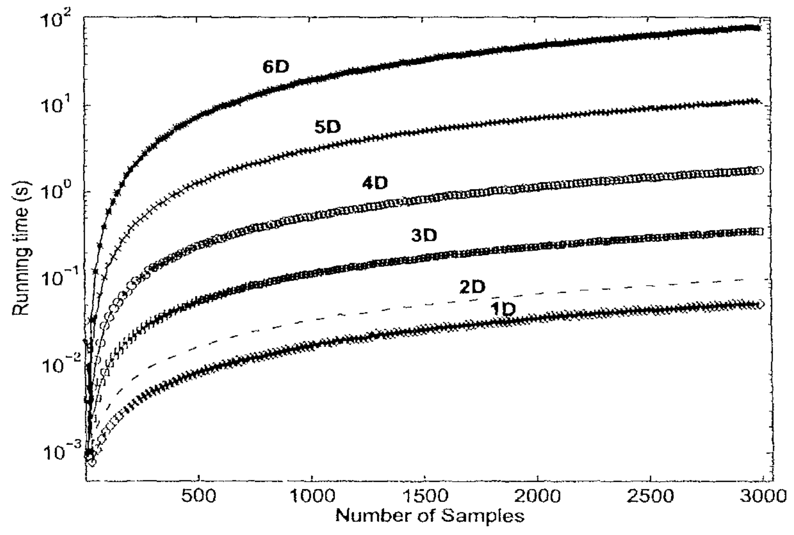


Figure 9.7: Time to complete a Delaunay triangulation using QHull (in seconds)

Figure 9.7 shows how the running time of QHull scales with the number of data

[5]Constructing an accurate global model typically involves an iterative process of collecting data →training models →collecting more data→updating models →etc. Each time data is collected is referred to as a sampling iteration.

points and number of dimensions. The tests were done on an Intel Quad core 2.6GHz machine with 4GB main memory. In the worst case (3000 points in 6 dimensions) the triangulation takes about 80 seconds which is still an acceptable cost to make once every sampling iteration. The plot also confirms that the dimensionality influences the running time the most.

Thus, since this cost need only be made once every sampling iteration, it is negligible for any but the high dimensional (>6D) applications of global surrogate modeling methods. For those cases switching to an iterative or approximate Delaunay implementation will provide a faster solution.

9.4 Applications

We now discuss two applications to demonstrate the usefulness of the LRM idea: one from electronics and one from structural dynamics. These problems were chosen since they are both challenging real world approximation problems from engineering, yet data generation is still sufficiently cheap to allow multiple runs and the generation of dense test grids for model validation purposes.

9.4.1 Low Noise Amplifier

First we revisit the RF circuit block modeling problem tackled in chapter 5. There the goal was to accurately capture the non linear behavior of a Low Noise Amplifier (LNA) while minimizing the number of adaptively selected data points. The general conclusion of the study was that ANN models gave the most favorable accuracy vs. samples ratio but that they were computationally expensive to generate using standard, re-sampling based model selection criteria. With the development of LRM we re-did some of the experiments and these are reported below.

9.4.1.1 Background

Recall from chapter 5 that obtaining the required circuit design parameters can be done through an approximation of the circuit performance figures based on one or more surrogate models. This is referred to as 'forward model' of the circuit. A forward model can be either obtained via direct modeling of circuit performances (i.e., a one step approach) or by using intermediate surrogate models of a convenient set of behavioral parameters (e.g. admittances and noise functions) and computing performances via analytical equations in a post-processing step (i.e., a two step approach). This is illustrated in figure 9.8.

Subsections 9.4.1.2 and 9.4.1.3 will test both approaches through the use of LRM. The purpose of subsection 9.4.1.2 is to validate the use and implementation of the LRM metric on a representative case from the study in [20]. The purpose of subsection 9.4.1.3 is to apply LRM to the new problem of modeling the performance parameters

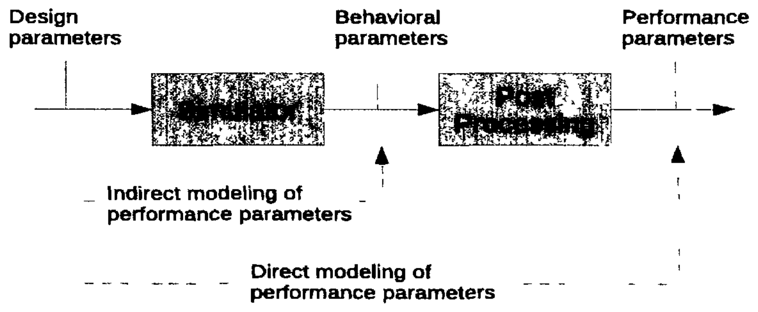

Design parameters
Behavioral parameters
Performance parameters
Simulator
Post Processing
Indirect modeling of performance parameters
Direct modeling of performance parameters

Figure 9.8: Direct and indirect modeling of the LNA performance parameters

directly and seeing if satisfactory accuracies can be reached (within the sample budget) as the dimensionality is increased.

9.4.1.2 Modeling of the circuit parameters

We first take the 4D formulation of the LNA problem and attempt to reproduce the behavior of the input-noise current response variable $\sqrt{i_{in}^2}$ defined by the same set of approximate equations as used in chapter 5.

Configuration The SUMO toolbox was configured with ANN models using a genetic algorithm to optimize the topology and the initial weights. The GA was run for 10 generations between each sampling iteration with a population size of 10. The networks were trained using Levenberg-Marquardt backpropagation in conjunction with Bayesian regularization [173, 272] for 300 epochs. Thus, note that we already apply a model specific regularization algorithm to help keep the smoothness of the model (and its behavioral complexity) under control. A good overview of regularization approaches in neural networks is given in [438].

The adaptive sampling algorithm used is LOLA-Voronoi [239] which determines the non-linear regions of the true response and samples those more densely. LOLA-Voronoi depends only on the true data and not on the surrogate model. LOLA-Voronoi starts from a small optimized Latin hypercube design augmented with the corner points of the domain. Each sampling iteration LOLA-Voronoi selects 30 new points up to a total of 1500. A different run is done for each model selection metric used: in-sample error, max-min 20% validation error, 5-fold cross validation, AIC, unbiased test set, and LRM. In addition each run was redone but this time with LRM as an auxiliary measure. In this case the average of both was taken. Each run was repeated 15 times to smooth out random effects. All tests were run on CalcUA, the cluster available at the University of Antwerp, which consists of 256 Sun Fire V20z nodes (dual AMD Opteron with 4 or 8 GB RAM), running SUSE linux, and Matlab 7.7 R2008b.

Results Figure 9.9 shows how the true error decreases under different model selection criteria as more data points are added. The true error is calculated as the Root Relative Square Error (RRSE) on a test set of 11^4 points.

The "true error" curve in figure 9.9 shows the error curve if the true error itself is used to drive the model generation. Intuitively this should be the best curve since it is the most accurate fitness function for the hyperparameter optimization. Note that in real situations the "true error" is unknown.

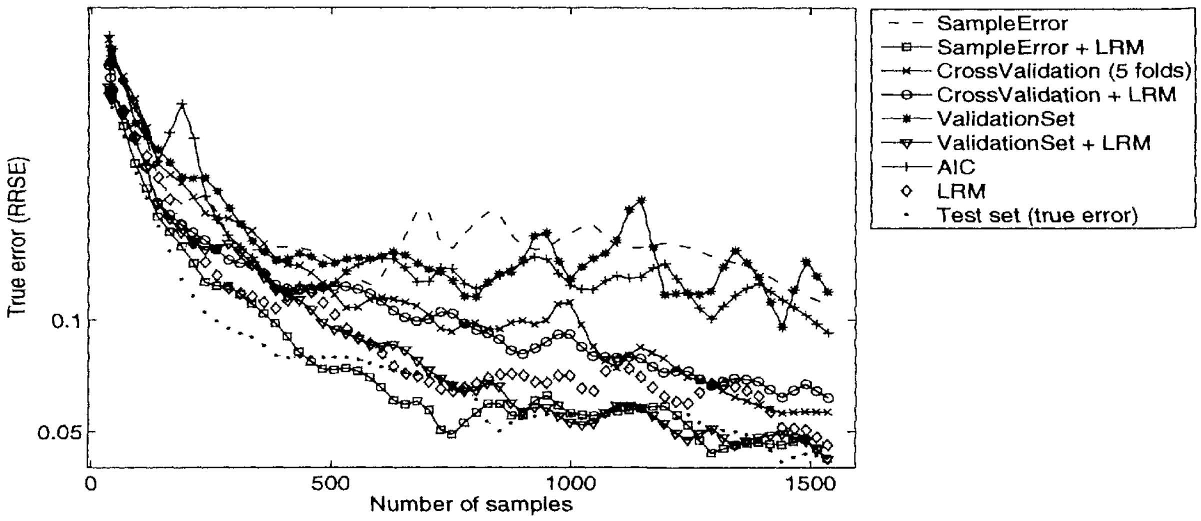


Figure 9.9: Ability of different model selection criteria to approximate the true error (LNA application)

A number of interesting observations can be made from the figure. Firstly, figure 9.9 shows using LRM alone is already a reasonable approximation for the true error, even outperforming 5-fold cross validation. Surprisingly, using no generalization control (Sample Error curve) performs just as well (poorly) as using a 20% max-min validation set or AIC. This is most likely due to the Bayesian regularization already employed in the ANN implementation. Note also how the evolution of the error tends to be more erratic in those cases.

While using LRM alone already gives reasonable results, the accuracy is improved further if it is used as an auxiliary measure. For example, when combined with the in-sample error its performance is roughly on par with with minimizing true error directly. The results are also comparable to those in [20]. Likewise, auxiliary use of LRM significantly improves the evolution of the validation set curve, but seems to have less impact on the cross validation curve.

If we then look at the structural (genotype) complexity of the models generated under the different measures we see that the results are again quite good. While LRM only acts on the phenotype complexity, figure 9.10 shows that it also impacts the genotype complexity favorably. Performance of pure LRM is similar to using cross validation

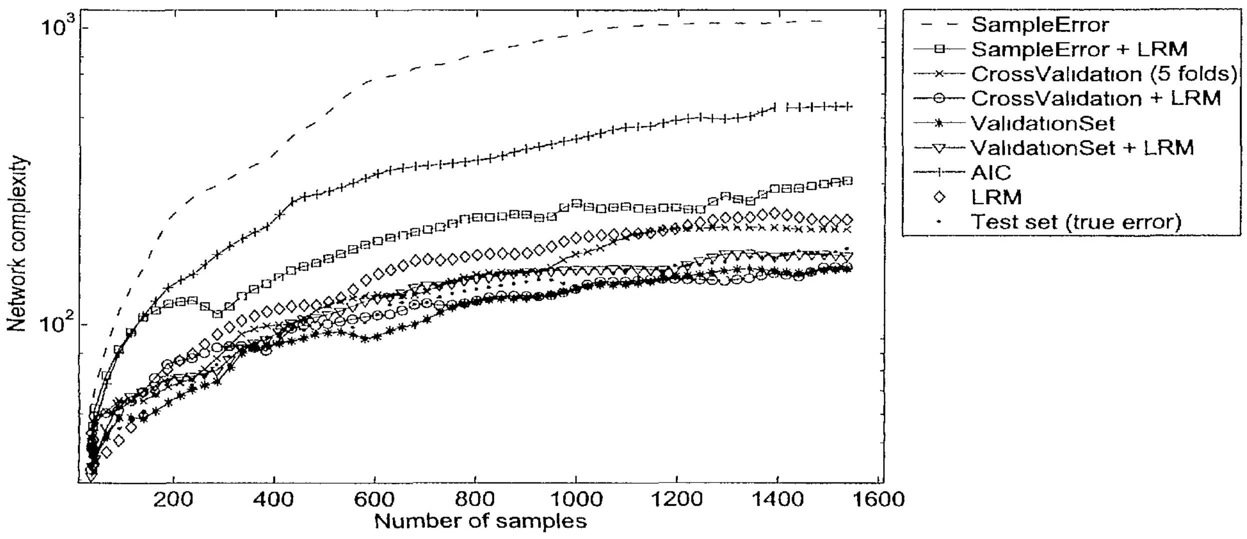


Figure 9.10: Evolution of the model complexity (# weights) under different model selection criteria (LNA application)

or the true error directly and results are further improved if it is used as an auxiliary measure.

The final question is then how this affects the overall running time. This is shown in figure 9.11. For reference, note that the training of a single ANN model in this case takes between 2 and 45 seconds (depending on the complexity) with an average of about 20 seconds in this case. This time is due to the Matlab implementation (vs. native code) and the expensive Bayesian regularization employed[6].

Again LRM performs competitively to using the true error directly and almost an order of magnitude faster than if cross validation were used. One may wonder about the strange *SampleError* curve. Since it is the fastest measure to evaluate one would not expect it to take the most time. The reason is that, while the measure itself is cheap, it leads to more complex models which, in turn, lead to longer training times.

9.4.1.3 Direct modeling of the performance parameters

The previous section was, like chapter 5, concerned with modeling the electrical behavior (admittances, noise parameters) based on low level circuit parameters. These models can then be used to derive the performance characteristics of the LNA circuit in a post-processing step (figure 9.8). However, recent work [444] has shown that small errors present in the intermediate surrogate models are amplified through the post-processing equations and can thus render the final prediction of the performance parameters unstable and inaccurate. An alternative approach is to model the perfor-

[6]In general, for these tests the modeling cost is offset as soon as the simulation time exceeds roughly one minute.

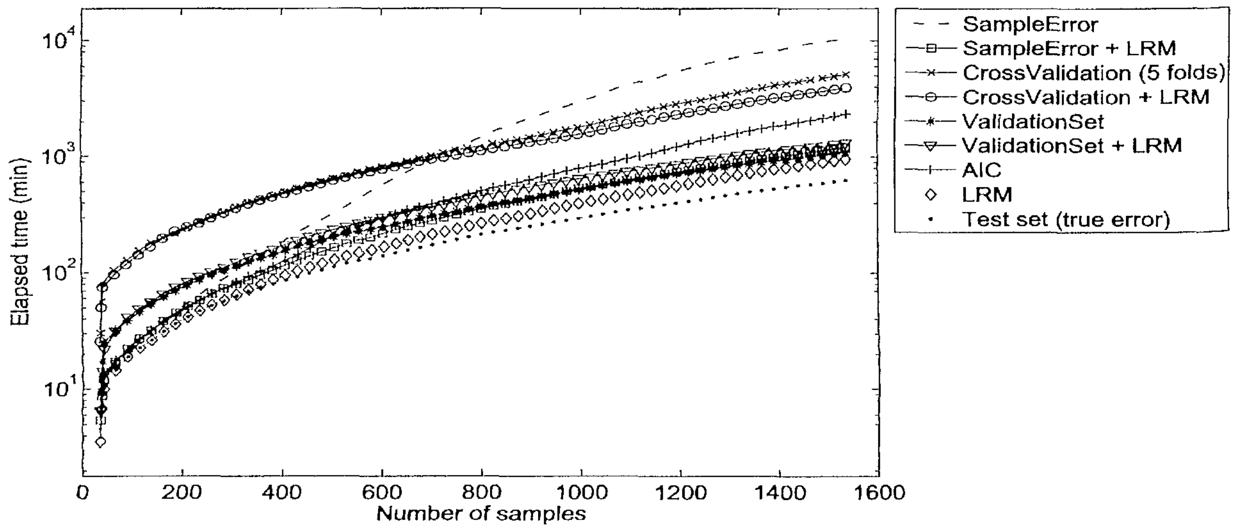


Figure 9.11: Comparison of the running time under different model selection criteria (LNA application)

mance parameters directly, without creating intermediate models of the electrical characteristics. Note that this is a completely different modeling problem with completely different response behaviors.

Configuration The previous subsection validated the use of LRM with respect to other model selection criteria. In this section we will use it as an auxiliary metric together with the in-sample error since the previous section showed it to be a good compromise. As relevant input parameters we take the transistor width W, the transistor length L, the source inductance L_s, the load resistance RL, the voltage bias of the transistor V_{GS}, and the resistance in series with the generator RS (the generator series resistance). As output parameter we take the second order nonlinearity $IIP2$. We vary the dimensionality from two to six and model $IIP2$ using the same setup as the previous section (the sample limit was extended to 3000). A full comparison across different model selection criteria (as done in the previous section) is not done for cost reasons. Rather we are interested to see if the use of LRM can lead to satisfactory ANN models within the sample budget and across the different dimensions. Reference test sets of size $51^2, 15^3, 11^4, 7^5, 5^6$ are available for this purpose. Otherwise the configuration settings were the same as for the indirect modeling case.

Results Figure 9.12 shows the curve for each number of inputs. The curves again depict how good the LRM-SampleError combination is at minimizing the error on the reference grid. Desirable features are a smooth, monotonic decrease of the error in function of the number of data points. The steeper the descent the better. Erratic jumps should be avoided but temporary increases in error are permitted. The error may temporarily increase if adding new data points reveals new features in the data or a new interpretation. What was thought to be a good model may turn out to be less accurate given the new information.

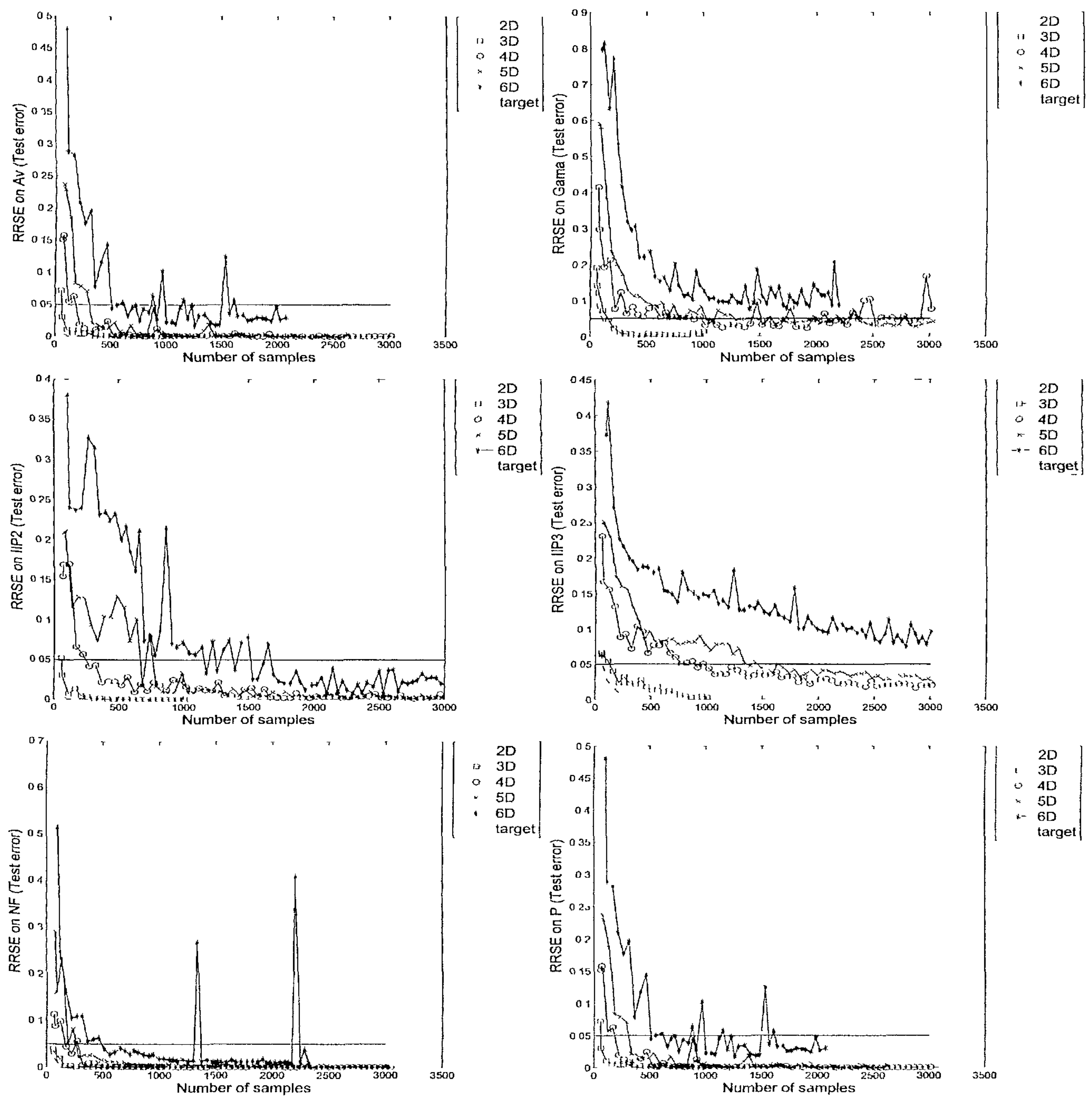


Figure 9.12: Evolution of the true error when using LRM and the in-sample error as the model selection criterion.

Studying the different sub-plots in figure 9.12, the first thing we see is that, besides

*IIP*3 and *Gama* for 6 inputs, all models reached the predefined targets[7] within the sample budget. A second observation is that the curves for 5D and 6D are rather erratic, much more so than the 2D-4D curves. This is confirmed by preliminary results on the other performance parameters [241]. The most likely reason for this lies in the fact that for more than 4 dimensions the number of LRM test points chosen per simplex is too small to allow for an accurate estimation of the magnitude $h(\mathbf{x})$. Note that 3000 points in 6 dimensions is only 3-4 points per dimension. The average simplex volume grows exponentially with the number of dimensions while the number of test points only grows linearly. Thus these results seem to imply that a different test point distribution is needed in more than four dimensions in order to more accurately guide the hyperparameter optimization. This is an issue for further research.

9.4.2 Truss

The second application comes from a completely different, but highly relevant, scientific domain: structural dynamics.

9.4.2.1 Background

The problem originates from [379] and is the geometric design of a two-dimensional truss for maximum passive vibration isolation. The baseline regular structure is shown in figure 9.13. This truss is a two dimensional simplification of a type typical in satellite applications. The structure is constructed of 42 Euler-Bernoulli beams, with two finite elements per beam, and is subject to a unit force excitation at node 1 across a 100-200Hz frequency range. The two leftmost nodes are encastre and all other nodes are free. The objective is to maximize the band-averaged vibration attenuation at the tip compared to the baseline structure. The geometry of the structure is varied by allowing nodes 1-20 to move inside 0.9×0.9 squares (as shown in figure 9.13). We consider a four variable problem with the x- and y-coordinates of node nine and ten as the variables and the other nodes fixed as per the regular structure.

9.4.2.2 Configuration

The same configuration was used for the truss application as for the LNA problem. Since the finite element analysis of this simple structure is very quick we were able to compute a 12^4 reference grid that is used to estimate the true error. We again use ANN models for this problem since initial tests showed they needed less points than other methods (SVM, Kriging, RBF) to achieve the same accuracy. Since they are also prone to overfitting and expensive to use they are a good example of where LRM is useful.

[7]Note that in general no reference grid is available. In that case whether a model is satisfactory or not will depend on (1) the engineer who visually explores the model, (2) the performance of the larger system where the global surrogate model is plugged into, and (3) a small number of extra test points.

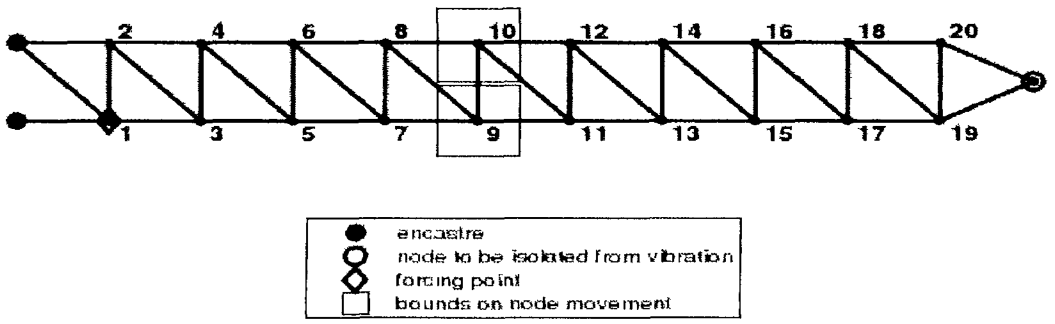

2
4
6
8
10
12
14
16
18
20
1
3
5
7
9
11
13
15
17
19
● encastre
○ node to be isolated from vibration
◇ forcing point
□ bounds on node movement

Figure 9.13: A regular truss geometry, showing node numbers, encastre points, the forcing point, and the two nodes to be isolated from vibration (adapted from [379]).

9.4.2.3 Discussion

Figure 9.14 shows how accurate each model selection metric is at approximating the true error. The results are closer together than for the LNA problem, though again we see that the LRM-SampleError combination performs very well, even slightly better than using the test set directly. The evolution of the cross validation, LRM, and validation set curves are essentially the same. Zooming in shows 5-fold cross validation performing slightly better than LRM-cross validation, followed by pure LRM, and then by the max-min 20% validation set curve. In this case using LRM as an auxiliary measure has little effect on cross validation and validation set. There is however a dramatic improvement when using it together with the in-sample error. Note the very poor performance of AIC. For this problem it punishes the complexity of the network too severely in order to be of any use.

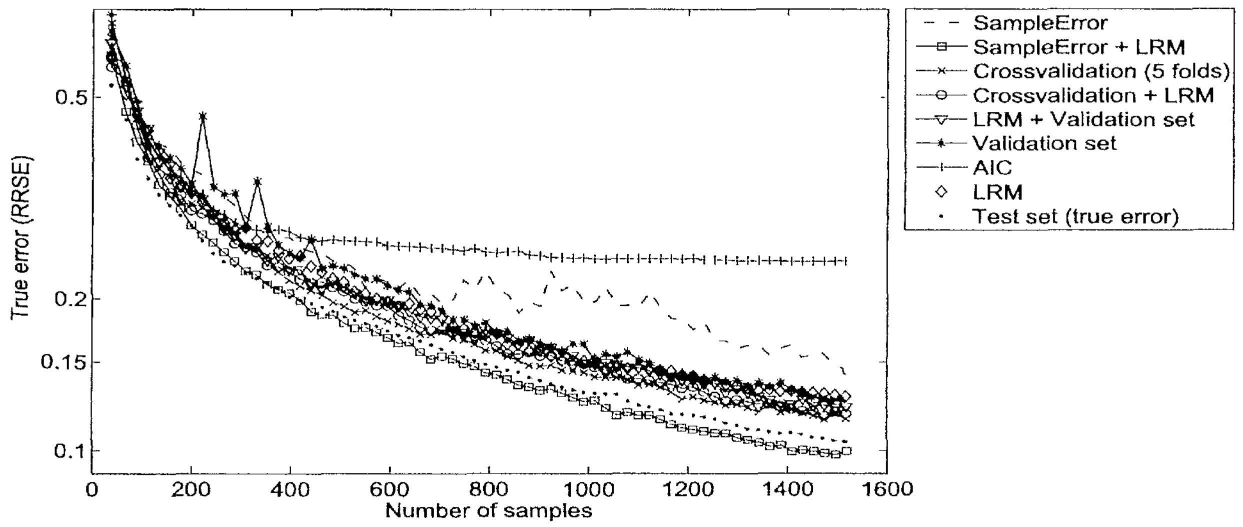


Figure 9.14: Ability of different model selection criteria to approximate the true error (Truss application)

Figure 9.15 shows how the different metrics impact the network complexity. Unsurprisingly, the use of the in-sample error alone again leads to the most complex networks. Adding LRM to the in-sample error significantly reduces the model complexity but it is still higher than using cross validation or the test set alone. On the other hand, using LRM alone results in the most parsimonious models (disregarding the AIC curve). More so than in the LNA example. LRM is a bit more conservative than cross validation when it comes to network complexity. This also seems to explain the small difference in accuracy between the two (figure 9.14). The tempering effect on the validation set curve can also be seen.

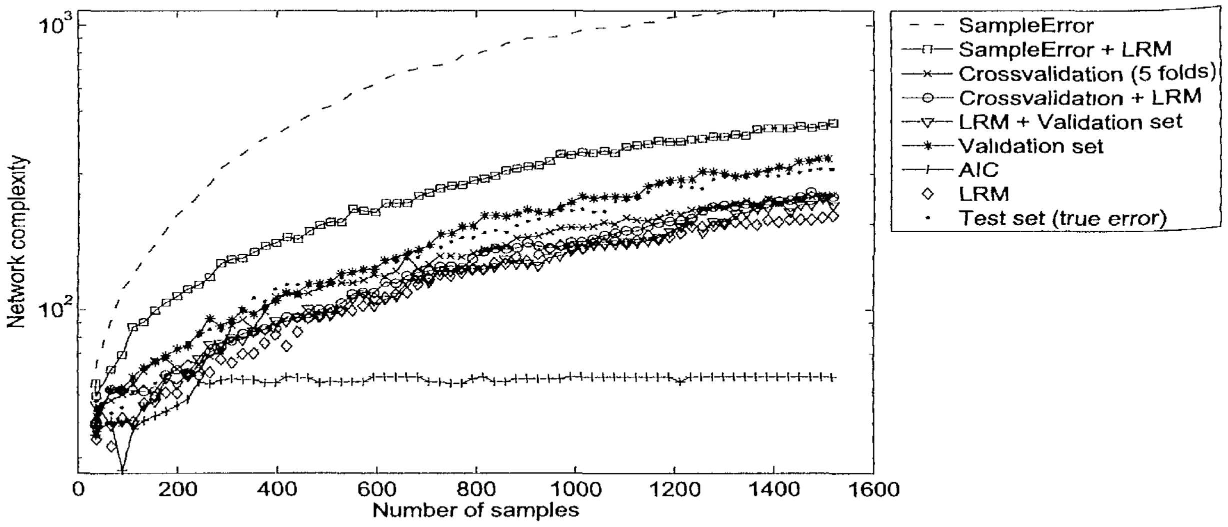


Figure 9.15: Evolution of the model complexity (# weights) under different model selection criteria (Truss application)

Finally the evolution of the running time across different metrics is shown in figure 9.16. The results are in line with those from the LNA problem. Using LRM alone is just as fast as using a separate test set, just over 4 times faster than 5-fold cross validation, and the parsimony pressure it applies makes using it much faster than using the in-sample error alone.

9.4.2.4 Summary

The motivation for LRM stemmed from the need for a faster replacement for cross validation that was just as generic and easy to apply, and was better at keeping the behavioral complexity of the model under control. Given the two applications, we see that LRM seems to have been able to achieve this, at least for the ANN models employed here. The accuracy curve is comparable or better than that of 5-fold cross validation with a lower model complexity and computational cost. It is also clear, as others have noted as well [439, 445], that AIC should not be used for ANN models. Rather a more model specific metric like the Network Information Criteria (NIC) [445]

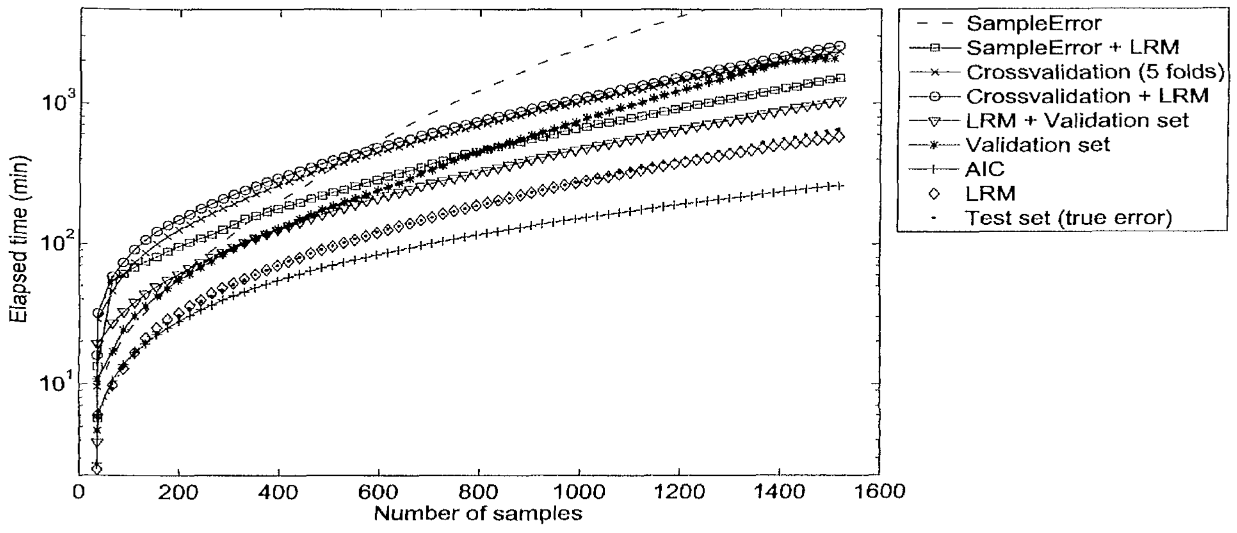


Figure 9.16: Comparison of the running time under different model selection criteria (Truss application)

should be used instead. This however requires changing the model implementation, something which we wanted to avoid.

9.5 Conclusion and Future Work

In this chapter we have discussed the motivation, algorithm, and application of a new model selection criterion (LRM) for estimating the response nonlinearity that can be useful in itself or combined with existing model selection criteria as a penalty. LRM was designed to directly combat the unwanted oscillations that can occur in fitted responses and serve (alone or combined) as a cheaper alternative to traditional resampling methods. Also useful is that its calculation does not require extra training or validation points and that an LRM value can be directly compared across different model types. This makes it possible to seamlessly apply it in hybrid methods such as [446]. However, again we stress that LRM should never replace existing (model specific) complexity control or regularization criteria. These should always be applied first. Rather, LRM should be applied (alone or as a penalty) where these are unavailable or insufficient.

Naturally more research is needed to investigate how much other model types benefit from LRM. Initial tests with RBF SVM models show similar improvements in accuracy and also show the stabilizing effect of LRM as a penalty with existing methods. Work is also underway on an iterative implementation and improved test point distribution scheme in order to improve scalability in more than six dimensions (driven by preliminary scalability tests in [241]). This would allow application to larger data sets or codes. How the LRM metric can be successfully applied in the complex domain (to allow its use for certain electro-magnetic problems) is also an open issue.

It turns out the LRM metric can also be reformulated as a sample selection criterion with some attractive properties. If a model shows a peak or oscillation in a particular area of the domain, the domain expert needs certainty that the peak/oscillation is actually there and not just an artifact of the model. This can only be guaranteed by sampling the peak/oscillations and verifying whether the model predicts the correct behavior. If this is not done the domain expert will tend to mistrust the model. The same is true for the different optima. The top of each optimum should be sampled to ensure that the height (or depth) of the peak is indeed correct. If LRM is used as a sampling criterion this is exactly what happens. This is a topic of ongoing work.

Various extensions to the basic LRM idea are also possible. For example one could also include the volume of the simplex in the LRM equation and thus give more (or less) weight to certain areas of the model. The same applies to the prediction uncertainty. Also, depending on available domain knowledge, the reference model need not necessarily be a linear one. Extension to other reference models leads to algorithms not unlike the various space mapping methods from the electronics community [74]. Extension of the LRM idea to these methods has the potential of further gains in accuracy.

10 Applications

A simulation saved is a simulation earned

– Anonymous

10.1 Introduction

In this last chapter we present a list of problems that were tackled throughout the work conducted for this dissertation. These problems add to the already diverse set of applications discussed in the previous chapters and come from such diverse domains such as hydrology and aerodynamics. In some cases more information and/or movies of the modeling process can be found on the SUMO Lab blog (`http://sumolab.blogspot.com`) and YouTube channel (`http://www.youtube.com/user/sumolab`).

10.2 Shekel function

10.2.1 Description

The first example is a synthetic problem where the objective is to model a predefined, 4-dimensional mathematical function. The function in question is the Shekel 5 function [447], a popular function for testing global optimization algorithms. This function was chosen since previous modeling results are available for comparison. In [16] the authors compare Second Order Regression, Kriging Models, and the Datascape modeling software[1] on three problems: Earth-Mars transit orbit design, the Shekel function, and Satellite constellation design. Only the Shekel function is used here since we did not have access to the code for the other problems.

The function definition for 4 variables is given by equation 10.1.

$$f(\mathbf{x}) = \sum_{i=1}^{5} \frac{1}{(\mathbf{x} - A_i)(\mathbf{x} - A_i)^T + c_i} \tag{10.1}$$

where

$$A = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{bmatrix} \quad \text{and} \tag{10.2}$$

$$\mathbf{c} = \begin{bmatrix} 0.1 & 0.2 & 0.2 & 0.4 & 0.4 \end{bmatrix}^T$$

10.2.2 Experimental setup

The Shekel function was modeled once with adaptive sampling switched on (with a maximum of 1000 samples), and once with it switched off. In the latter case the problem was modeled multiple times for different fixed Latin Hypercube designs (in accordance with [16]): 25, 50, 100, 250, 500, 1000, 2500, 5000, and 10000 samples respectively. In the former case an initial optimized Latin hypercube design of size 20 is used augmented with the corner points. Modeling is allowed to commence once at least 90% of the initial samples are available. Each iteration a maximum of 10 new samples are selected using the LOLA-Voronoi adaptive sampling algorithm. As in [16], a Latin Hypercube design of size 20000 is available for validation.

The SE is modeled once with adaptive sample selection and the automatic model type selection algorithm, and once without sampling (for different fixed experimental designs) using feed-forward ANNs. ANNs were used for the second test since they scale well with the large number of data points needed.

For the first test the model types included are: RBF models, ANNs, Kriging, Rational models, and LS-SVMs. In accordance with [16] the following error functions are used:

R-Squared (R^2):

$$R^2(\mathbf{y}, \tilde{\mathbf{y}}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{MSE}{Variance} \tag{10.3}$$

[1] Datascape models have been used on development and operations of the F-16, which saved an estimated \$36 million [16].

Average Absolute Error (AAE):

$$AAE(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{\sum_{i=1}^{n} |y_i - \tilde{y}_i|}{n} \tag{10.4}$$

Maximum Absolute Error (MAE):

$$MAE(\mathbf{y}, \tilde{\mathbf{y}}) = \max(|y_i - \tilde{y}_i|) \text{ for } i = 1, .., n \tag{10.5}$$

where $\mathbf{y}, \tilde{\mathbf{y}}$ are the true and predicted response values respectively. The R^2 value on the validation set is used to drive the hyperparameter optimization.

The SUMO Toolbox was set to terminate if the number of samples exceeds 1000 (sampling enabled) or a maximum of 5 hyperparameter optimization iterations is reached (sampling disabled).

10.2.3 Results

Figure 10.1 shows the results from [16] together with those from the SUMO Toolbox. It is immediately clear that the performance of the latter is very good. In the adaptive modeling case (ANN models) the error decrease in function of the number of samples is larger than for the other techniques. For example, to reach a maximum absolute error of 2, Datascape needs 5000 samples, while the ANN models generated by the SUMO Toolbox need only 1000.

The difference is even more marked if we consider adaptive sampling. Here RBF models were selected as the best fitting type. If samples are selected iteratively and models are updated sequentially, only 275 samples are needed to obtain a MAE of 2. After 1000 adaptively selected data points (roughly 5.6 per dimension) the final RBF model has a MAE of 0.13 (1% relative) and a AAE of 0.0024 (< 1% relative). Which is still less than the other model types reach after 10000 data points. In addition, in the case of 10000 samples, the average time to build one Datascape model is 2 hours, vs. 30 minutes for RBF models[2], and 1-2 minutes for ANN models. The final RBF model after 1000 samples is shown in figure 10.2.

Remark that these results should not be taken as a formal full fledged comparison since not enough information is available about the other methods. The methodology used in Datascape is proprietary, thus we cannot comment on reasons for difference in accuracy. All that is mentioned is that Datascape uses features from fuzzy logic, non-linear regression and numerical optimization and presents them in a hybrid format [16]. In the SUMO case results were obtained with out-of-the-box default settings. No problem specific customization or user interaction was performed, this will of course only further improve the accuracy.

[2]Due to the large number of data points the RBF models were fit with an iterative Alternating Projections method.

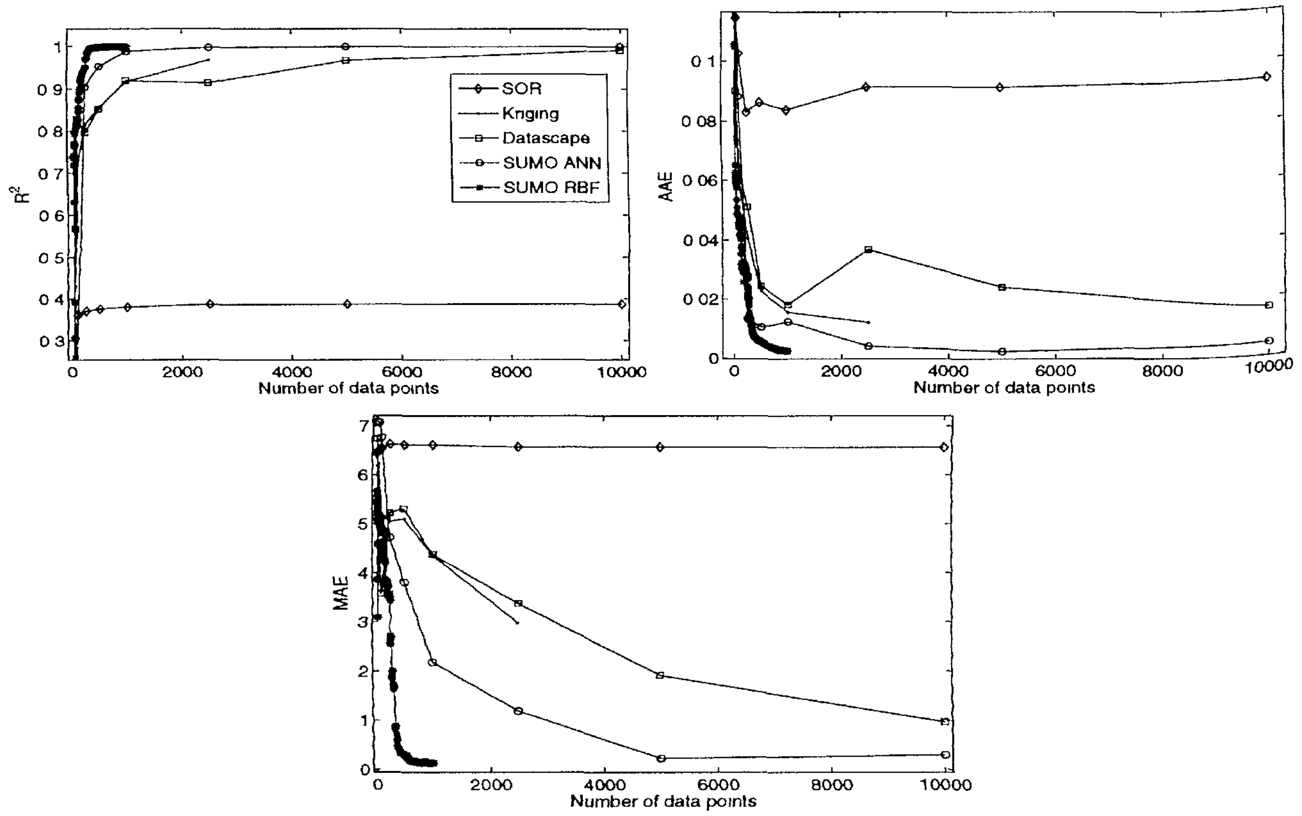


Figure 10.1: Evolution of the errors for different methods when modeling the Shekel function (R^2: left, AAE: right, MAE: bottom).

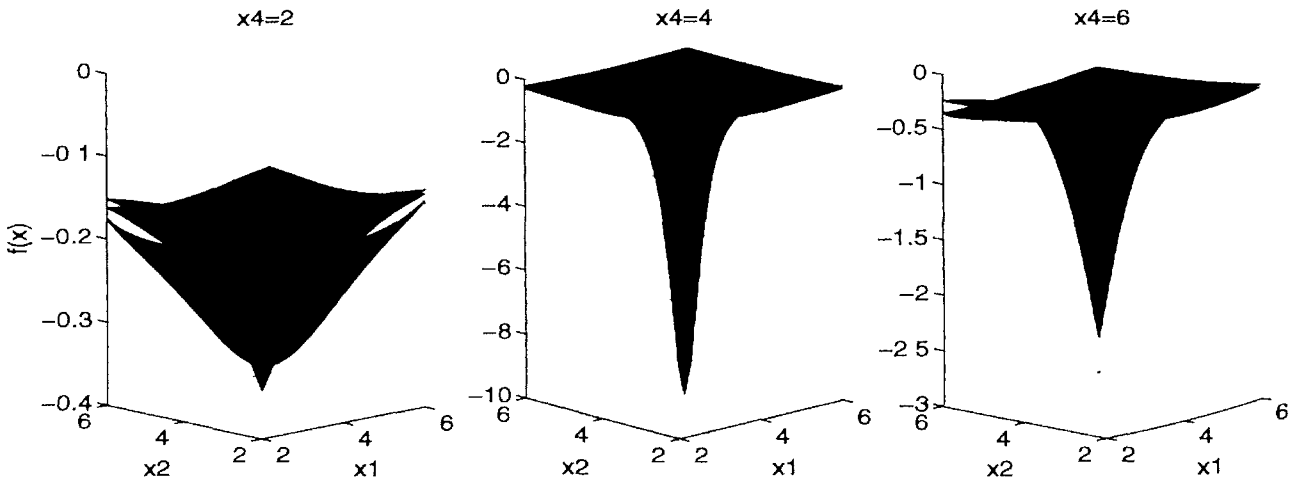


Figure 10.2: Final RBF model after 1000 samples (SE, x_3 shown at the minimum (red), maximum (green), and middle value (blue)).

10.3 Catchment parameter modeling

10.3.1 Background

We now take a modeling problem from hydrology. A task which is often central to hydrological modeling is the identification of suitable parameters for a given set of modeling objectives, catchment characteristics and data. However, this identification process is difficult because conceptual rainfall runoff models generally have a large number of parameters and the accuracy of their calculations depends on how the relevant parameters are defined. Additionally, because of their conceptual nature, these parameters cannot be measured directly and are therefore estimated on the basis of a calibration process, i.e., minimizing an objective function.

We illustrate the strength of global surrogate modeling in improving the process of estimating the right parameters of a rainfall runoff model. The SWAT (Soil Water Assessment Tool) is an operational model that was developed to assist water resource managers in assessing water supplies and non-point source pollution at river basin scale. The model is able to assess the impact of changes in climate, landuse and management, and to simulate the transport and fate of chemicals and water quality loadings. The model is designed so that use can be made of readily available inputs. Upland components include hydrology, weather, erosion/sedimentation, soil temperature, plant growth, nutrients, pesticides, and land and water management. Stream processes include channel flood routing, channel sediment routing, nutrient and pesticide routing and transformation. The ponds and reservoirs component contains water balance, routing, sediment settling, and simplified nutrient and pesticide transformation routines. Water diversions into, out of, or within the basin can be simulated to represent irrigation and other withdrawals from the system. However, one should be aware that every process in the model is a simplification of reality.

In SWAT, a watershed is divided into multiple subwatersheds, which are then further subdivided into *hydrologic response units* (HRUs) that consist of homogeneous landuse, management, and soil characteristics. The HRUs represent percentages of the subwatershed area and are not identified spatially. The model operates in a continuous mode and has been widely used to estimate catchment runoff, nutrient and sediment loads. The SWAT model development, operation, limitations, and assumptions are extensively discussed by [448]. One of the practical problems in applying the SWAT is determining proper values for the more than 30 parameters that control the fidelity of its prediction. While many parameters can be estimated empirically a direct expensive optimization procedure is still routinely used to determine optimal settings [449], requiring many expensive simulations.

Through the use of sequential modeling and active learning methods, a replacement metamodel can be generated that captures the relationship between the different SWAT parameters and provides insight in their influence on the prediction quality of the SWAT. While at the same time minimizing the number of computationally expensive simulations. Optimization can still be performed as a post-processing step.

10.3.2 Related work

A few studies have been reported in recent years in the field of water resources related to surrogate modeling. Savic et al. [450] applied 2 data-driven models (genetic programming and ANN) to flow prediction, results show that both are able to match up against conceptual models. Khu et al. (2003) [451] reduced the number of simulation runs required by Monte Carlo (MC). This was achieved by using an ANN and hybrid GA to respectively approximate and explore the shape of the objective function. This significantly reduces the computational effort involved in investigating hydrological model parameter uncertainty. Later on, an evolutionary-based metamodel calibration methodology was developed using a coupled genetic algorithm-RBF ANN [452]. Regis and Shoemaker (2004) [354] proposed an approach for costly black-box optimization that uses space-filling experimental designs and k-nearest neighbor local function approximations to improve the performance of an EA in twelve-dimensional groundwater bioremediation problem. Broad et al., (2006) [453] evaluated six local search algorithms for purpose of improving the performance of ANN surrogate model-based optimization of water distribution systems. The results show a significant improvement in the value of the objective function by using a local search as a complementary stage of surrogate model-based optimization of water distribution systems. Kamali et al. (2007) [454] evaluate the performance of the design and analysis of computer experiments (DACE) surrogate function along with Latin Hypercube Sampling (LHS) and MC Sampling for hydrological model calibration. The results indicate that DACE along with LHS reduced the computational cost of the calibration process. Recent research by Garote et al. [455] advocate the use of Bayesian networks to learn the behavior of a rainfall runoff model.

10.3.3 Experimental setup

10.3.3.1 SWAT

The SWAT requires spatial information about topography, river/stream reaches, landuse, soil and climate to accurately simulate the streamflow. The study basin is that of the Grote Nete (383 km^2), located in the north-eastern part of Belgium. A detailed description of the study basin is given in [456]. Daily observations of precipitation, air temperature, evaporation, and daily streamflow data were obtained from the Royal Meteorological Institute and the Flemish Administration for Land and Water, Belgium. The soil map was available at a scale of 1:25.000; the soil physical data was derived from the Aardewerk-SIBIS Soil Information System and land use was derived from the multi-temporal LANDSAT 5 TM image of 18 July 1997.

The climatic inputs in SWAT include daily precipitation measured in 5 stations scattered in and outside the study area, and the potential evapotranspiration and min/-max temperature collected in a station at the northern boundary of the catchment. Details of input data are given in [456]. The catchment was subdivided in 8 subcatchments and 65 HRUs. The flow separation program of [457] was used in this research as to determine the relative contribution of surface runoff and groundwater to total streamflow. The latter were created based on the various combinations of land use and soil types present in the catchment. Climate data were assigned to each HRU using the centroid method. The daily streamflows in the Varendonk outlet station were used for model calibration and verification.

Parameter sensitivity analysis was applied to identify the parameters of the SWAT model that contribute most to the variability of component flows. It is important to have an understanding of catchment characteristics and the hydrological processes involved before 'blindly' applying surrogate modeling to the available data. Based on a critical analysis of the SWAT modules to the hydrology of the study area, the parameters to calibrate were reduced to 18. Although this number of parameters is considerably smaller, to further reduce the number of parameters in the surrogate process, a sensitivity analysis was conducted to determine the most sensitive parameters of the hydrological module simulating streamflow. This analysis (through Latin Hypercube and One-factor-at-a-time) yielded the 4 most sensitive parameters.

The first parameter is p, the percentage by which CN_2 (the Soil Conservation Service (SCS) curve number) is changed from the initial values. Thus, p, a parameter in the approximation model, is converted to CN_2, the actual parameter of the SWAT, using the following formula: $CN_2 = initial CN_2 + \frac{initial CN_2 \cdot p}{100}$. Secondly, $RCHRG\_DP$ stands for the deep aquifer percolation ratio and is a measure for the transfer between the shallow and deep aquifer system. Thirdly, $REVAPMN$ is the amount of water (mm) that must be present in the shallow aquifer store before water can move to the unsaturated zone. Finally, $ESCO$ is the soil evaporation compensation coefficient. The domains of the 4 parameters are [-40,40] (ensuring absolute bounds of [35 90] for CN_2), [0 3], [0 1] and [0 1] respectively. When the SWAT model is run it generates a time series of predicted flow during the period 1998-2002. This time series is then separated into 3 components useful for runoff prediction: *low flow* (values ≤ 2), *high flow* (values ≥ 5), and *total flow* (all values). On each of these components the Mean Square Error (MSE) is then calculated with the true observations during that period, and that is the final output of the simulation code. Separating the total flow in more fine-grained components allows the SWAT to be calibrated for different types of flows. Thus, in sum, the SWAT simulator has 4 inputs (CN_2, $RCHRG\_DP$, $REVAPMN$, $ESCO$), and 3 outputs ($MSE_{low}, MSE_{high}, MSE_{total}$).

10.3.3.2 SUMO Toolbox

The active learning settings were set as follows: an initial optimized Latin hypercube design of size 50 is used augmented with the corner points. Modeling is allowed to commence once at least 20 of the initial samples are available. Each iteration a maximum of 50 new samples (over all outputs) are selected using the LOLA-Voronoi adaptive sampling algorithm [239] up to a maximum of 500.

There are many surrogate modeling methods available to fit the data and many options implemented in the SUMO Toolbox. However, from the application it is not immediately clear which surrogate model type or hyperparameter optimization algorithm should be used (ANN, SVM, RBF models, ...). For this reason we shall use the automatic surrogate model type selection algorithm from chapter 7. The surrogate model types included in the evolution are: single layer feed forward ANNs (using [413]), Kriging models (using [270]), rational functions and LS-SVMs (using [269]). Together with hybrid models (ensembles, that arise as a result of a crossover between two models of different type) this means that 5 model types will compete to fit the data. The population size for each model type is 10 and the maximum number of generations between each sampling iteration is 15. The final population of the previous model type selection run is used as the initial population for the next run. An extinction prevention algorithm is used to ensure no model type goes completely extinct. Given the correlation between the outputs, they are not modeled separately (by separate models) but together in a single model with multiple outputs.

Note that this approach relieves the domain expert from technical choices related to the model generation. Besides a few high level options (which model types are of interest) and termination criteria (time limit, sample budget) no further input is required. The hyperparameter optimization, model selection, and sample selection are performed fully automatically, allowing the domain expert to concentrate on the application and not have to deal with modeling technicalities.

In order to drive the hyperparameter optimization a max-min validation set of 20% is used. Since not all data points are available at once but are chosen incrementally, the validation set grows as more data arrives. Validation points are not selected randomly but by maximizing the minimum distance between them, thus ensuring a good coverage of the domain. Note, though, that models are always trained on all the data, it is only when the error is calculated that they are temporarily re-trained on 80% of the available data. The error function that is minimized is the Average Relative Error (ARE): $ARE(\mathbf{y},\tilde{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i-\tilde{y}_i|}{|y_i|}$, where $\mathbf{y}$, $\tilde{\mathbf{y}}$ are the true and predicted response values respectively. Since we are dealing with multiple outputs per model, a weighted sum over the ARE values for each output is taken. Since we wish to treat all outputs equally, all weights were set to 1[3].

The SUMO Toolbox was configured to use the remote Sun Grid Engine (SGE) sample evaluation backend (see section 6.4). This means that the toolbox will run simulations in parallel by transparently submitting them to a remote cluster. The cluster in question is the CalcUA cluster which consists of 256 nodes. Thus the SUMO Toolbox is running on a local machine, while the SWAT simulations are scheduled on the cluster. The number of data points selected each iteration is chosen dynamically (but never exceeding the user defined limit of 50) based on the average time needed for modeling, the average duration of a single simulation, and the number of compute nodes available

[3]Alternatively, a multi-objective approach as discussed in chapter 8 could also have been used.

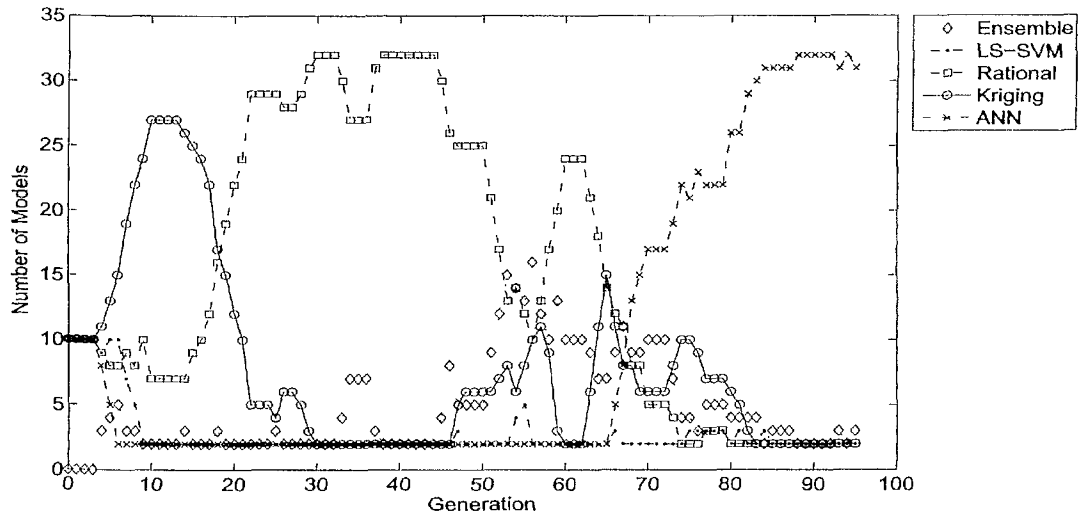


Figure 10.3: Evolution of the population composition during the modeling of the catchment parameters.

at that point in time. The average time for one simulation is quite short, 4-10 minutes depending on cluster availability.

10.3.4 Results

Figure 10.3 shows the evolution of the population as the modeling progresses. Some interesting dynamics can be observed. As soon as migration between the different sub populations is allowed to take place, Kriging models quickly take over the population resulting in very smooth approximation surfaces. As the number of data points increases, the quality of the rational functions increases and they overtake Kriging as the most popular model type. However, the problem with the rational functions is that they are very prone to producing asymptotes in their response due to the increasing existence of poles. The implementation in the toolbox is best suited to low dimensional cases with sufficient data per dimension, in other cases the orders of the polynomials involved grow too quickly, increasing the risk of overfitting. Therefore, it is no surprise that they are finally overtaken by ANN models that, thanks to the pruning functions implemented as part of the mutation and crossover operators, are able to produce smoother responses.

Of course nothing prevents this process from recurring. The fact that the optimal solution changes with time is not necessarily a bad thing and should actually be expected since the hyperparameter optimization landscape is dynamic (due to the active learning). Note that it is the extinction prevention algorithm that makes these oscillations possible (it ensures a model type never goes completely extinct but that at least 2 individuals of each type are preserved). Without extinction prevention these dynamics are impossible and everything depends on the initial conditions. As a result the danger of converging to a poor local optimum (poorly fitted regression model) is significantly larger.

$\|X\|$	*Output*	*Training error* (*ARE*)	*Validation error* (*ARE*)	*Cross validation*	minimum x^*	$f(x^*)$
500	MSE_{low}	0.083	0.108	0.104	(-39.99, 0.69, 0.99, 0.65)	0.83
	MSE_{high}	0.025	0.036	0.028	(-37.04, 0.00, 0.00, 1.00)	15.63
	MSE_{total}	0.023	0.038	0.028	(-39.62, 0.30, 0.99, 0.67)	10.38

Table 10.1: Errors of final ANN model (4-14-3 network)

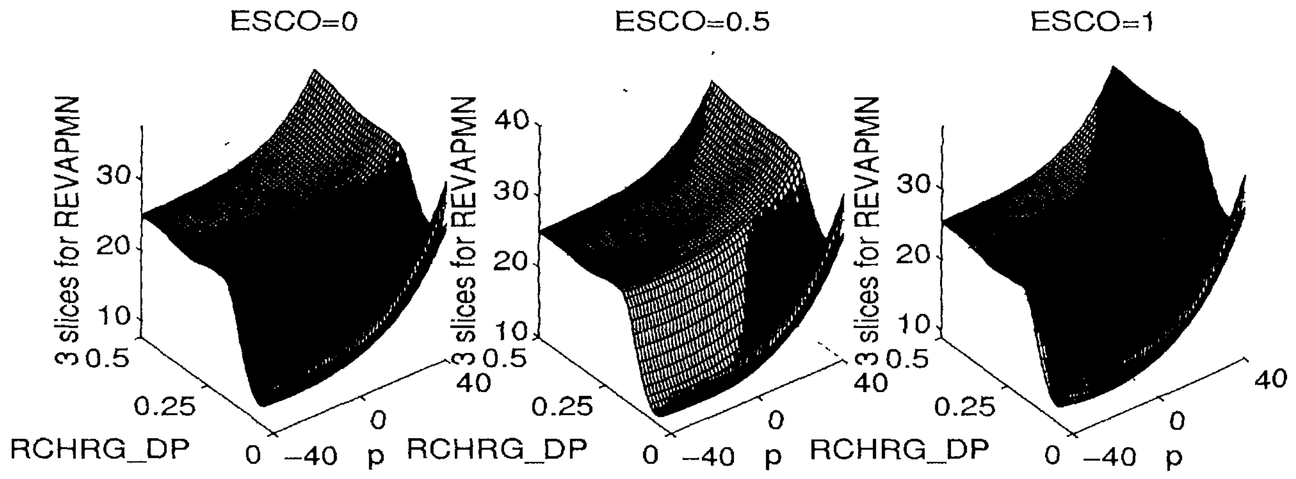


Figure 10.4: Final ANN model for MSE_{total}, plotted at the minimum, maximum, and middle values of REVAPMN. The fact that the three slices coincide indicates REVAPMN has a negligible impact on the overall response.

Table 10.1 shows the final average relative errors (ARE) for each of the outputs on the training and validation data. In addition a 10-fold cross validation error was calculated as well. $|X|$ is the number of samples used to train the ANN model while x^* and $f(x^*)$ denote the minimum and corresponding function value of the ANN model respectively. For the MSE_{high} and MSE_{total} an error of less than 5% (acceptable for the application) is easily reached. The MSE_{low} output appears more difficult, reaching only a final *ARE* of 10%. Thus future runs should take this into account, placing more emphasis on the first output instead of treating all outputs equally. On the other hand, this can also be an indication that the hydrologic model parameters selected are not good enough to capture the trend to simulate base flow. Therefore incorporating more parameters like available water capacity of soils (*SOL_AWC*) will improve not only low flow simulated values but also high flow simulated values. This is the topic of a follow-up publication.

Figure 10.4 shows the plot of the final best model (a 4 – 14 – 3 ANN) for MSE_{total}. In the figure *REVAPMN* and *ESCO* have been clamped at 3 values: 0, 1 and 3 for *REVAPMN* and 0, 0.5 and 1 for *ESCO*. The remaining 2 parameters, p and *RCHRG_DP*, are shown along the x-axis and y-axis respectively.

From the figure it is immediately clear that the 3rd and 4th parameters have virtually no influence on the quality of the SWAT prediction: the three slices of each subplot coincide and the three subplots for each output show little or no differences. This was confirmed by using the model browser of the SUMO Toolbox to browse through each of the 4 dimensions. This is an unexpected result, a further study of the study basin and HRU settings is underway to shed more light on this issue. Though, a preliminary explanation can be given as follows. The 3rd variable, *REVAPMN*, affects when and to what degree subsurface flow occurs, and therefore indirectly govern the contribution of subsurface flow to the total stream flow of the watershed of interest. These two parameters (*ESCO* and *REVAPMN*) have more influence in evapotranspiration simulated by the model. Since we just analyze flow simulated by the model, these values cause a non-noticeable change in the water yield calculations, and therefore adjustments to these values can be left out.

Interesting is also the break point $RCHRG\_DP = 0$, below which the quality of the SWAT prediction markedly improves, reaching a minimum of 0.8 (MSE_{low}), 16 (MSE_{high}), and 10 (MSE_{total}) respectively. The models also clearly show that the SWAT has more trouble predicting high flows than low flows (as can be seen from the higher MSE_{total} value). Peak flow predictions were generally appreciable for low events and poor for higher flow rates because SWAT uses a modified formulation of the Soil Conservation Service (SCS) curve number (CN) technique [458] to calculate surface runoff. This result is consistent with earlier findings that the SWAT tends to overestimate peak flows [459]. In sum, the model captures the relationships between the different parameters in a smooth and intuitive manner.

10.3.5 Conclusion

In this section the computationally expensive problem of parameter setting in rainfall runoff modeling was investigated (calibrating the SWAT). The final surrogate model produced by the SUMO Toolbox provided insight into the relationship between the different parameters (including identification of the optima) and can be used to improve the prediction quality in other settings (e.g., as part of a wider Geographic Information System (GIS) tool).

10.4 Methane-Air Combustion modeling

10.4.1 Background

This example and its description is taken from [389], where the authors describe the generation of an optimal ANN using a pattern search algorithm.

The chemical process under consideration describes methane/air combustion. The GRI 2.11 chemical mechanism containing 277 elementary chemical reactions among 49 species is used. The steady laminar flamelet equations [460] are often employed to describe the reaction-diffusion balance in non-premixed flames. The solutions to these equations provide temperature and mass fractions of all species in terms of two parameters. The mixture fraction z and the reaction progress variable c are used for this parametrization. Temperature and the chemical source term of c, which can be viewed as a measure of heat release, are shown as functions of these parameters in figure 10.5. The solid lines represent the system boundary and flame states outside this boundary are inaccessible. It can be seen that the chemical source term is very localized around $z \approx 0.2$ and $0.15 \leq c \leq 0.23$.

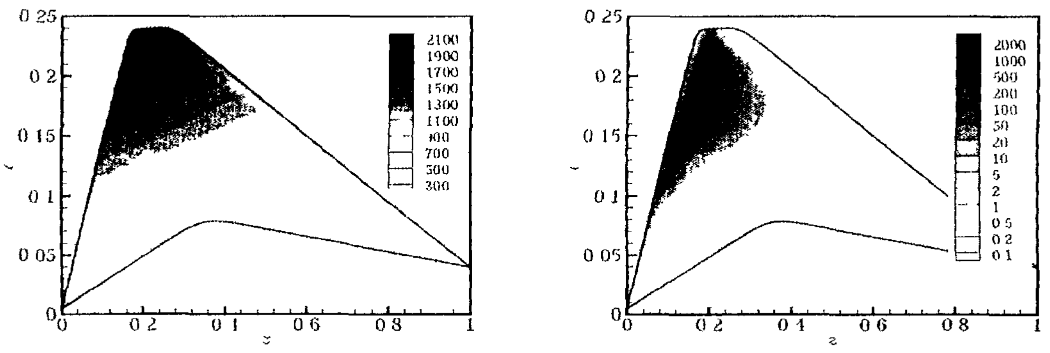


Figure 10.5: Solution (from [389]) of the steady laminar flamelet equations as a function of mixture fraction z and progress variable c; (a) temperature (K) and (b) chemical source term (kg/$(m^3 s)$)

For the approximation 1000 data samples are available, half of which will be used for training, the other half for validation (in accordance with [389]). Sample data were obtained by applying an acceptation-rejection method [461]. This method consequently results in a better resolution of the important regions with high chemical source term and temperature. In addition to the training and validation sets, a separate dense data set of 13959 samples is available for validation purposes. This data set it is *not* used in any way during the modeling but as a simple post-processing step to objectively show the accuracy.

10.4.2 Experimental setup

Since a fixed dataset is available, the adaptive sampling loop was switched off. Only adaptive modeling was performed. Again, it is not immediately clear which surrogate model type should be used. For this reason we use the automatic surrogate model type selection algorithm based on heterogeneous evolution. The surrogate model types included in the evolution are: feed forward ANNs, smoothing splines, RBF models, Rational functions and LS-SVMs (using [269]). Together with the hybrid models (ensembles) this means that 6 model types will compete to fit the data. The population size for each model type is 10, as is the maximum number of generations between each sampling iteration.

The validation set of 500 points is used to drive the model parameter optimization (as done in [389]) using error function (7.5). The SUMO Toolbox is set to terminate after timeout of 360 minutes or an accuracy of 0.01 is reached. Only the temperature output is considered here since it is the most interesting. In addition, the temperature data is normalized to $[0, 1]$ and the input space is normalized to the interval $[-1, 1]$.

10.4.3 Results

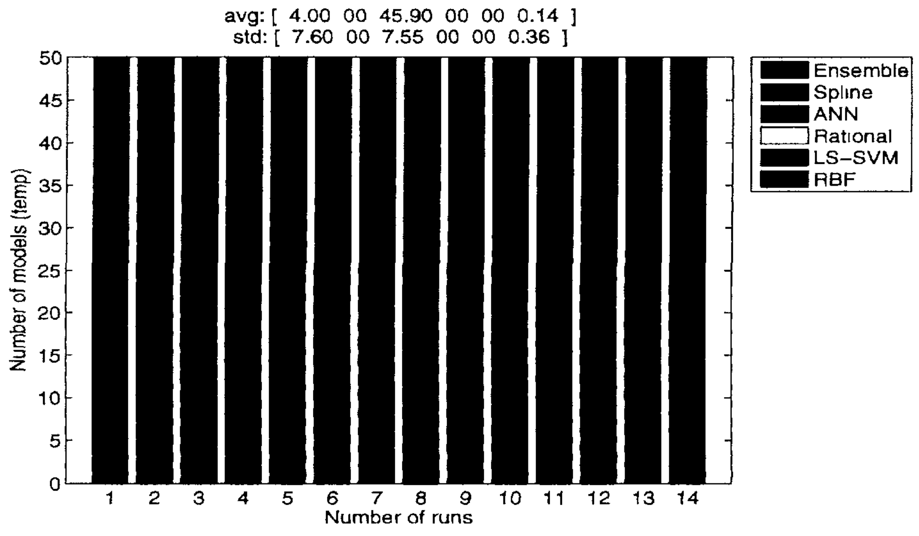
Each experiment was repeated 14 times. The composition of the final population for each of the 14 runs is shown in figure 10.6a.

From the figure it is clear that ANNs perform the best in all cases and that the results are consistent (the average and standard deviation are shown at the top of figure 10.6a, the leftmost value corresponding to the top legend entry). The first run of figure 10.6a is interesting given the large number of ensembles. In that particular case the algorithm has found a combination of 2 or more ANNs that perform similar to or better than a single ANN. As an example of the population evolution, figure 10.6b shows the trace of a single run (run 9). The figure shows clearly that as soon as migration occurs ANN models quickly take over the population.

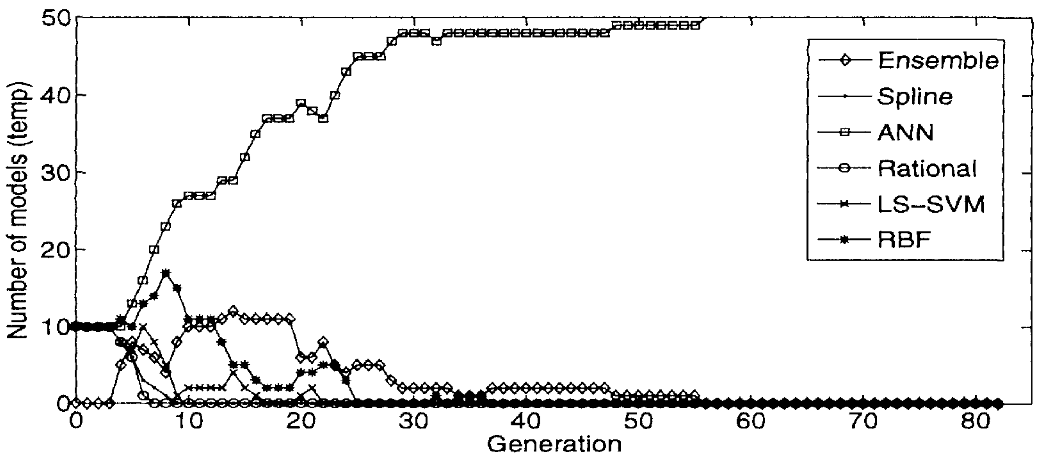
Figure 10.7a shows the generalization errors of the final best model found in each run. Each bar shows the error histogram of the final best model for each run on the independent test set (lighter is better). The evolution of the error histogram for one particular run (run 9) is shown in figure 10.7b (lighter is better). Both plots show that the generalization error is very good: an error of less than 0.01 on roughly 90% of the unseen data, and less than 0.1 on the remaining 10%. Remember that usually such test data is not available, it is simply used here to illustrate the accuracy of the model and was not used during the modeling in any way. A plot of the final best model for run 9 is shown in figure 10.8.

The model shown is an ANN with two hidden layers (size 8 and 9) and 120 free parameters (weights). The complexity was determined automatically from a starting complexity of 3 units in each layer.

Finally, table 10.2 shows the different errors averaged over all 14 runs. The table



(a) The composition of the final population in each run.

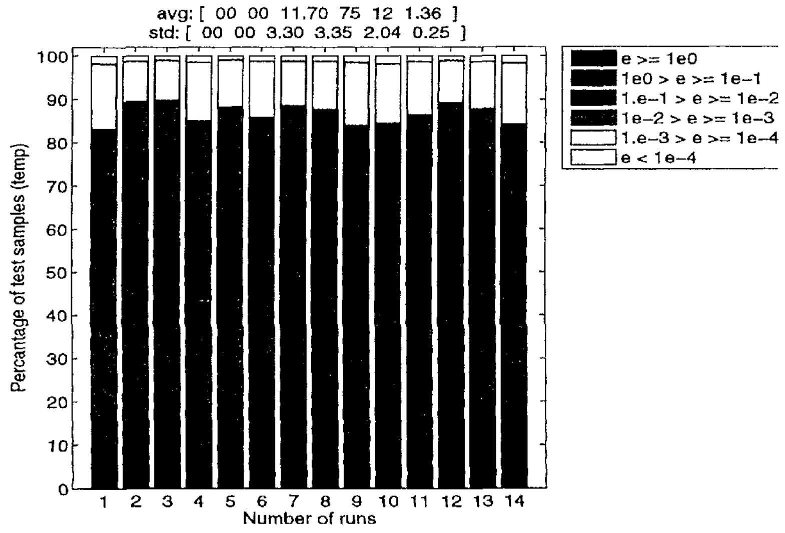


(b) The evolution of the composition for the 9th run.

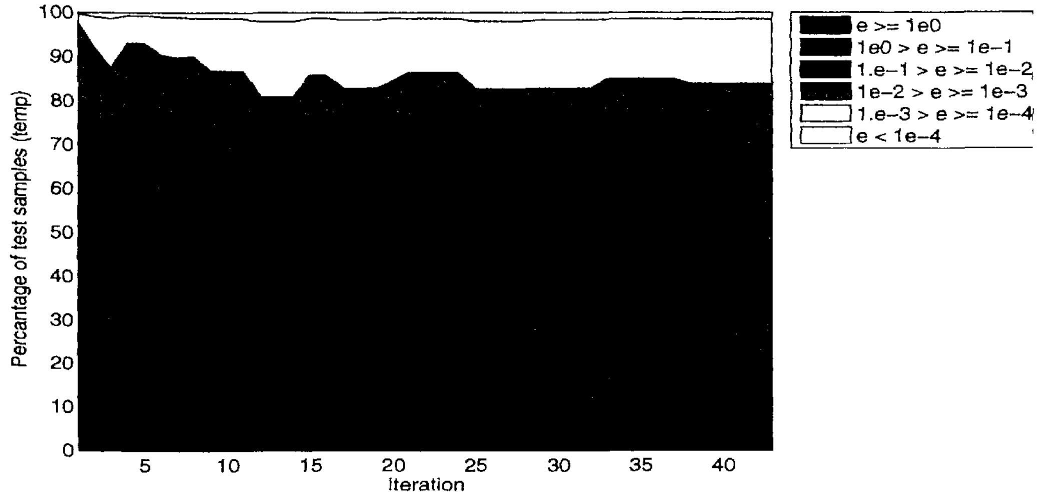
Figure 10.6: Model type selection results for the combustion modeling problem.

also includes the results of running the different surrogate model types standalone. The table shows the *RRSE* on the validation set (the fitness value) and the *RRSE* and MRE_a on the test set respectively. The average running times are shown as well.

Table 10.2 shows that, despite the somewhat odd data distribution, all model types can fit the data reasonably well. However, only ANNs are able to reach the target accuracy of 1%. Comparing the standalone ANN results with the automatic model type selection we see that the differences are negligible. The automatic algorithm performs just as well as the best single model type run, at the cost of doubling the running time (due to the much larger population size, 50 versus 10). The advantage, however, is that the larger population size allows for more robust performance (cfr. the lower value of σ for the running time) and that the population size for each model type



(a) Test error histogram of the final model in each run.



(b) The evolution of error histogram for the 9th run.

Figure 10.7: Accuracy results for the combustion modeling problem.

varies dynamically, proportional to the quality of fit. Remember, though, that in any real application the modeling cost will still be largely outweighed by the simulation cost.

10.5 Airfoil design example

10.5.1 Background

The next example is an application from aerodynamics. It is based on "Airfoil Geometry Design for Minimum Drag" by Z. Wang [462] and is a nice example of airfoil design. [462] is concerned with finding an optimal design of a wheel fairing (or wheel

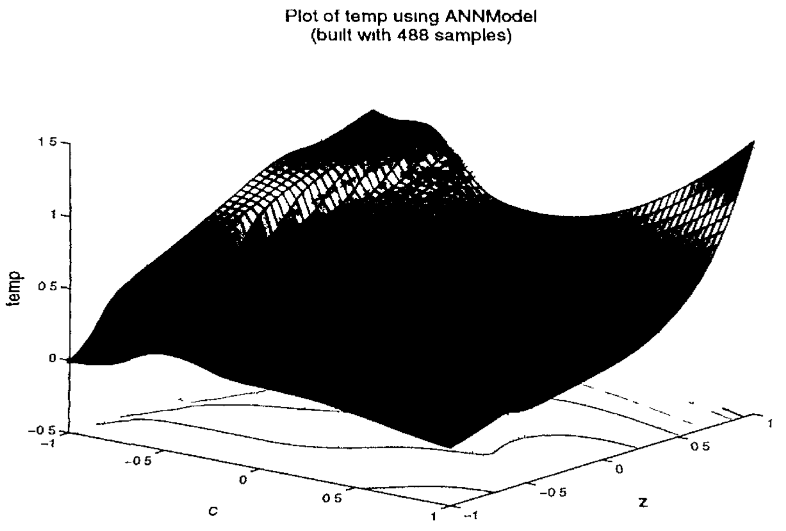


Figure 10.8: Normalized temperature plot of the final ANN model (CE, run 9)

Type	*Time (min)*	σ	*Validation* error (*RRSE*)	σ	*Test error* (*RRSE*)	σ	*Test error* (*MRE*$_a$)	σ
Spline	360	140	4.20e-02	1.62e-02	5.42e-02	3.83e-03	4.25e-02	9.04e-04
ANN	133	107	**9.85e-03**	3.86e-04	**5.12e-02**	1.13e-03	**3.92e-02**	8.92e-04
Rational	360	0	3.00e-02	0	2.23e-01	0	3.73e-01	0
LS-SVM	360	0	2.97e-02	7.50e-04	7.01e-02	5.37e-03	5.88e-02	9.50e-03
RBF	360	0	2.16e-02	5.68e-04	5.63e-02	3.41e-04	4.08e-02	2.98e-04
Automatic	274	74	1.01e-02	1.09e-03	5.21e-02	2.01e-03	3.96e-02	1.04e-03

Table 10.2: Comparison of the final accuracies achieved by each model type (combustion modeling problem).

pant) on a solar powered vehicle. When designing airfoils, the typical goal is to optimize the lift-to-drag ratio $\frac{Lift}{Drag}$, i.e., design an airfoil that has a high lift while having not a too high drag coefficient. However for a wheel fairing for a vehicle or airplane the main aim is to have a low drag coefficient C_d. There are several published standard airfoils, for example the NACA series of the, now dissolved, National Advisory Committee for Aeronautics [463] who created series of airfoils for different purposes using analytical equations (see figure 10.9 for an example profile). Nonetheless, in many cases it is useful and more efficient to create a custom-made design.

In this case Xfoil [464] is used to evaluate the performance of custom airfoils, while Matlab is used to construct different airfoil geometries searching for the optimum design, and calculating the objective function. The original optimization problem is defined as:

$$min_{Airfoil\,Geometry} Drag(Airfoil\,Geometry)$$

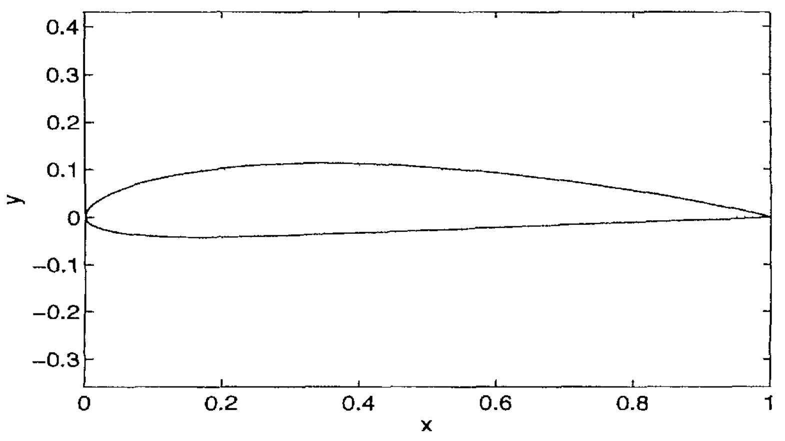


Figure 10.9: The NACA 4415: A wheel fairing airfoil [463].

subject to

$$Desired\,Thickness - Thickness(Airfoil\,Geometry) \leq 0$$

where the actual objective is constructed as $Drag(AirFoil\,Geometry) = C_d^0 + C_d^3 - k\,min(pressure^3)$. C_d^i denotes the drag coefficient at degree of attack i, $min(pressure^3)$ is the minimum pressure at degree of attack 3 and k is a weighting constant set to 0.00200. The original goal of [462] was to design a wheel fairing airfoil that would perform better than a NACA four digit airfoil. The following Xfoil options were used:

- Viscous mode is on
- Reynold number is 10^6
- Maximum number of iterations is 50 (viscous-solution iteration limit)
- There is auto point accumulation to active polar

Technically it works as follows. Initially, an airfoil geometry is generated with the spline toolbox of Matlab (using 4 control points). A discretization of this spline is saved to a file. Subsequently, this file, along with Xfoil instructions, is fed into Xfoil which simulates wind flow (computational fluid dynamics) and returns several performance metrics saved in a file. This file is easily read into Matlab which is able to combine the drag coefficient and maximum pressure into the aggregated objective mentioned above.

Summarizing the whole setup, there are 4 inputs (x,y,t_1,t_2) and 1 output *Drag*. The first 2 input parameters implicitly define 2 control points with coordinates (x,y) and $(x,-y)$. The other two control points are endpoints of the airfoil and are fixed at $(0,0)$ and $(0,1)$ respectively. The last 2 parameters are the tangents of these endpoints. This is illustrated in Fig. 10.10.

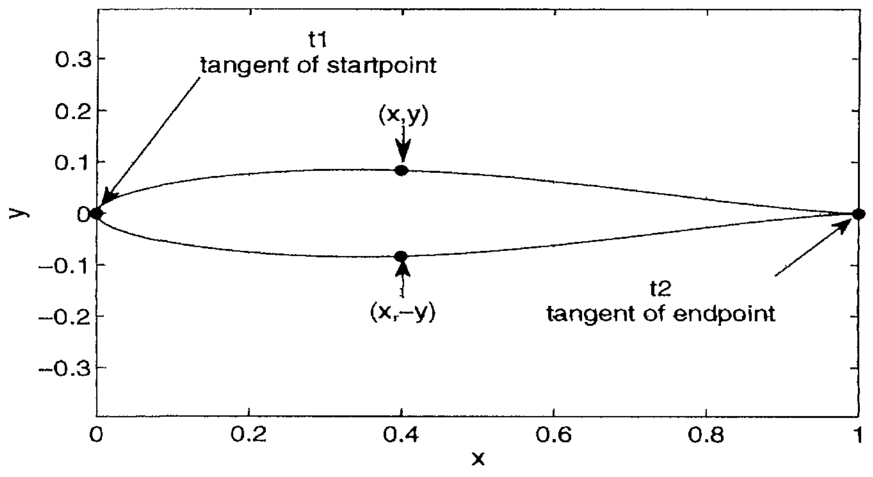


Figure 10.10: Optimal airfoil geometry.

However, in some cases knowing the optimum is not sufficient. Instead global information (relationships between parameters, sensitivities, ...) is needed. Using a global surrogate a designer is able to explore the design space in a more efficient manner, directly locating interesting designs, and gaining more insight into the behavior of the system.

10.5.2 Experimental Setup

The airfoil geometry problem has been modeled with the SUMO Toolbox using the same XFoil setup and objective function as [462]. The additional SUMO toolbox configuration was chosen as follows: A Latin Hypercube design of 20 points is generated and added to the 18 corner points to obtain 38 initial samples. New samples are adaptively chosen by the LOLA-Voronoi algorithm up to a maximum of 500. An ANN model (using the same settings as the previous example) is used to approximate the data. The hyperparameter optimization is driven by the RRSE on a max-min validation set of 20% which grows adaptively as more samples become available. Outliers caused by failed simulations (complete failure or wrong results due to failed convergence of the solver) were removed during sampling.

10.5.3 Results

Figure 10.11 shows the plot of the final ANN model. The final model is a 4-14-2-1 network with a RRSE of 0.01 on the validation set and (when the final model is trained on all the data) a 10-fold cross validation error of 0.17. Thus the attained accuracy is quite good. This is understandable, as Fig. 10.11 shows, the response is very smooth, almost parabolic, and thus easy to fit (this was further confirmed by using the SUMO model browser GUI). The model also shows the impact of the tangent t_1, t_2 parameters to be small. The three slices for t_1 almost coincide, so we can safely say that the tangent at the start point of the airfoil has virtually no effect on the combined drag. Similarly, the second tangent parameter t_2 only influences the drag for 'thick' airfoils (large values of x).

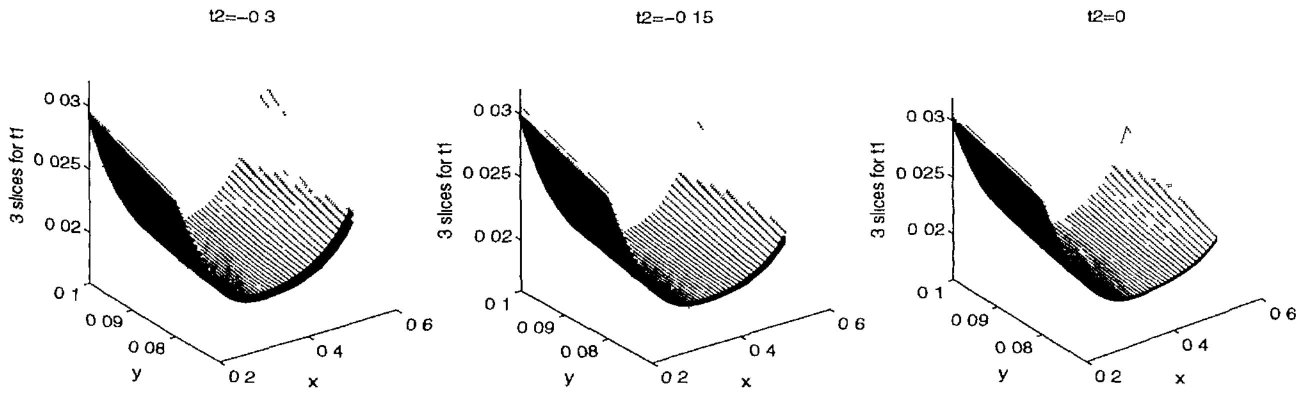


Figure 10.11: Final 4-14-2-1 ANN model of Drag (t_1,t_2 are plotted at 3 fixed values: the minimum, maximum, and middle value.

Besides a global model, the optimum is still of interest. Therefore, as a post-processing, the model was optimized (this can now be done very cheaply) using the DIviding REctangles (DIRECT) [465] algorithm. This resulted in a *Drag* value of 0.0166 at $x = 0.4, y = 0.0833, t_1 = 0.9167, t_2 = -0.05$. A plot of the airfoil geometry at this optimal point is shown in Fig. 10.10.

10.6 Metallurgy data

10.6.1 Background

As part of a collaboration with a large steel company a neural network model was constructed for data related to the composition of produced steel. Unconventionally, the problem here was not that data was scarce but that it was too abundant. There was too much data to be able to work with efficiently. A model was needed to summarize the data in an efficient, analytical model.

10.6.2 Results

Since the available dataset was so dense, it was possible to adaptively sample from the dataset directly and grow the neural network accordingly. The surface the needed to be fitted was far from trivial, yet the neural models were able to capture it very accurately, as can be seen from figure 10.12.

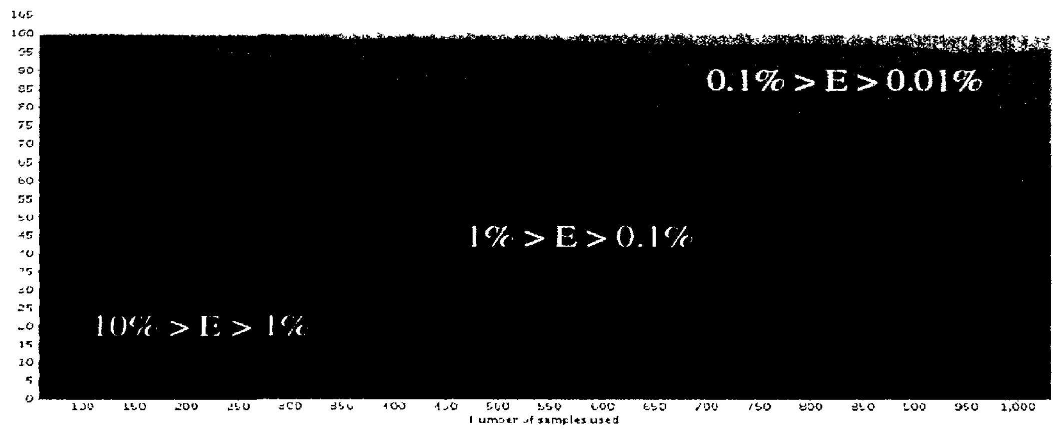


Figure 10.12: Evolution of the error histogram (on a test set) during the adaptive sampling and adaptive modeling procedure (Metallurgy problem).

10.7 Inductive Posts

10.7.1 Background

The next example is a 3D Electro-Magnetic (EM) simulator problem [373] (see also section 7.8.3). Two perfectly conducting round posts, centered in the E-plane of a rectangular waveguide, are modeled, as shown in figure 10.13. The inputs to the simulation code are: the signal frequency f, the diameter of the posts d, and the distance between the two posts w. The outputs are the complex reflection and transmission coefficients S_{11} and S_{21} (S_{12}, S_{22} are ignored due to symmetry). The simulation model was constructed for a standard WR90 rectangular waveguide with $f \in$[7 GHz, 13 GHz], $d \in$[1 mm, 5 mm] and $w \in$[4 mm, 18 mm].

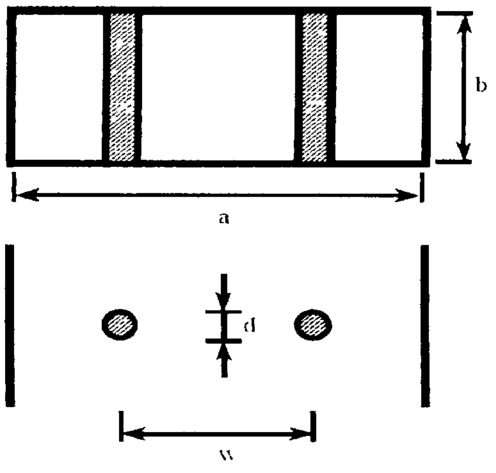


Figure 10.13: Cross sectional view and top view of the inductive posts (from [373])

For validation purposes an independent test set evaluated on a dense grid is available. It is used as an objective measure to gauge the accuracy of the final model. It is *not* involved in the modeling process in any way, instead cross validation is used. Obviously, in a 'real' setting such an objective measure is not available, we just use it here to illustrate that the modeling actually works.

10.7.2 Experimental setup

For the EM example the settings were as follows: an initial optimized Latin hypercube design (constructed using the method described in [242]) of size 20 is used augmented with the corner points. Modeling is allowed to commence once at least 90% of the initial samples are available. Each iteration a maximum of 10 new samples are selected using the LOLA-Voronoi adaptive sampling algorithm.

We use rational models since they are most adequate for this type of problem: the transfer function of the inductive posts system is a rational function and the final model can be used to check if EM laws are violated (e.g., passivity). Also, they are able to model complex valued data directly. This is not possible with other approximation methods like ANNs and SVMs[4]. For such methods the complex data must be split into real/imaginary or phase/amplitude parts (at the cost of an accuracy, performance, and interpretation penalty).

The hyperparameters of the rational functions (order of the polynomials, which variables should be in the nominator/denominator, etc.) will be optimized using a custom stochastic hill climber (though a GA or PSO algorithm could be used as well) [274]. In this case 5-fold cross validation is used as the model selection measure. The error function that is minimized is the root relative square error (*RRSE*):

$$RRSE(\mathbf{y},\tilde{\mathbf{y}}) = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{10.6}$$

With $\bar{\mathbf{y}}$ the mean true response. Intuitively the *RRSE* indicates how much better an approximation is than the most simple approximation possible (the mean). In addition we also record the adjusted Maximum Relative Error (MRE_a)

$$MRE_a(\mathbf{y},\tilde{\mathbf{y}}) = \max\left(\frac{|y_i-\tilde{y}_i|}{1+|y_i|}\right) \tag{10.7}$$

on the test set. Note that the MRE_a degrades to the maximum absolute error for small values of $|y_i|$.

The SUMO Toolbox terminates if the number of samples exceeds 500 or the accuracy drops below 0.02.

[4]Complex versions of ANN training algorithms have been proposed but numerous problems still need to be overcome before they can approach the power of their real-valued counterparts. The authors are unaware of complex formulations of SVMs.

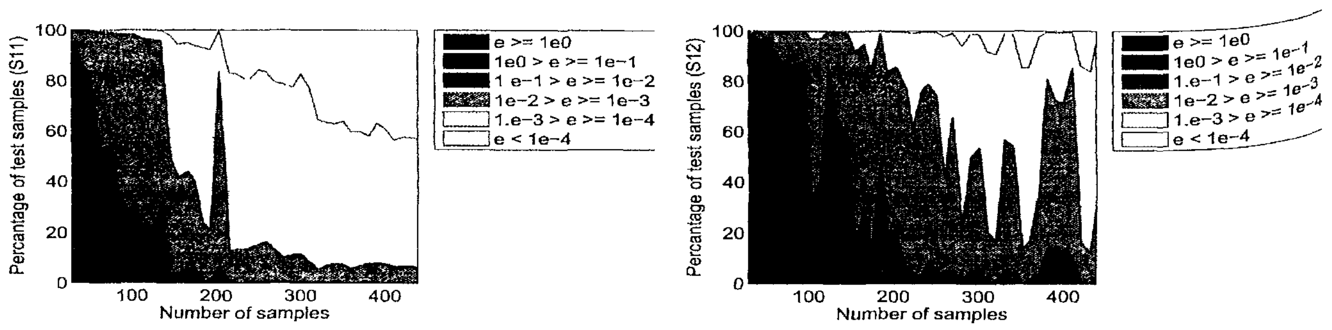
10.7.3 Results

The final errors for each output are shown in table 10.3 and the plot of the magnitude of the final best models is shown in figure 10.15. Table 10.3 shows the 5-fold cross validation error and the number of samples used ($|X|$). The evolution of the test error histogram is given in figure 10.14 (the lighter the region the more test samples with low error).

Output	$\|X\|$	*Cross validation error (RRSE)*	*Test error (RRSE)*	*Test error (MRE$_a$)*
S_{11}	438	5.79e-03	3.90e-03	7.03e-02
S_{12}	438	1.82e-02	1.51e-02	6.39e-02

Table 10.3: Accuracies of the final models for the inductive posts example.

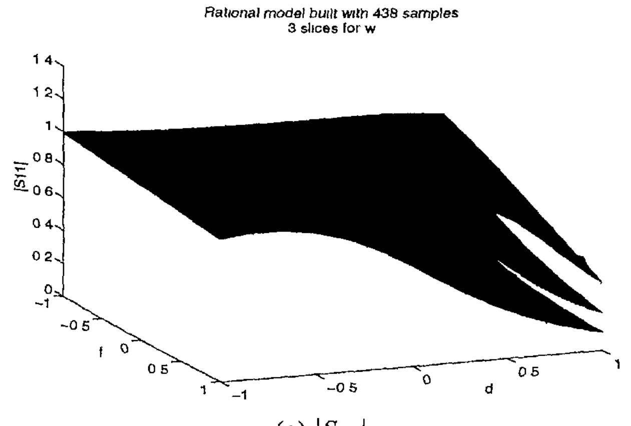
Evaluating the results, we see that the accuracy targets for both outputs are reached after 438 data points ($\approx$7.5 per dimension). Note that this number may be lower if both outputs are modeled separately. Since S_{11} is easier, its accuracy target is reached after just 233 samples. Regarding the test error histogram plots in figure 10.14, we see that the final generalization error is very low but that progress is rather erratic. This is to be expected since we are not using an exact measure to drive the hyperparameter optimization process. The cross validation error is only an approximation of the true error.



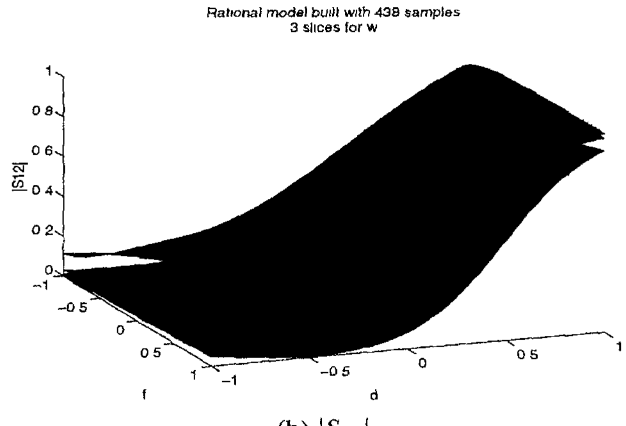
(a) S_{11} (b) S_{12}

Figure 10.14: Evolution of the test error histogram (Inductive posts example).

As a final remark, the cross validation errors given in table 10.3 are the global, averaged values but say nothing of the uncertainty of the model over the domain. Resampling strategies like the cross validation procedure, the .632+ bootstrap, or Jackknife, can also be used to produce an estimate of the prediction uncertainty at an arbitrary point. This can be done as a post-processing step to increase confidence in the final model. An alternative but similar approach is to combine a diverse collection of models into an ensemble and use their disagreement as a measure of uncertainty. In the case of the SUMO Toolbox this can be done by using the k best models generated



(a) $|S_{11}|$



(b) $|S_{12}|$

Figure 10.15: Normalized plot of the final rational model for the inductive posts example (w is clamped at -1 (blue), 0 (green), 1 (red)).

by the toolbox (with $k > 1$), instead of using just the best model. Another, more complex, alternative is to apply Bayesian Multi-Chain Monte Carlo (MCMC) methods to generate credible intervals and hyperparameter uncertainty bounds. References covering these topics (including model specific error propagation metrics) can be found in [466–468].

10.8 Double Folded Filter

10.8.1 Background

The third and final application from electronics is concerned with the problem of modeling a parametrized double-folded microstrip stub bandstop filter [229]. The filter is shown in figure 10.16.

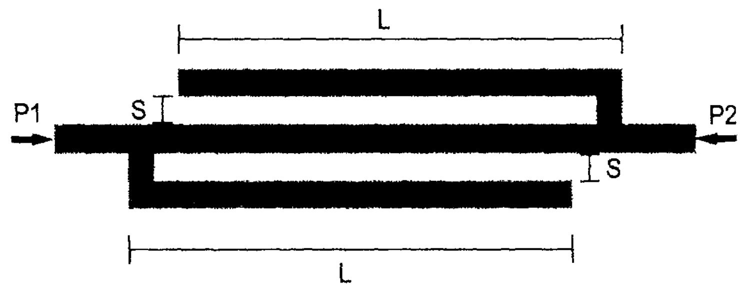


Figure 10.16: Double-folded microstrip stub bandstop filter

The substrate is 0.1270 mm thick with a relative dielectric constant $\varepsilon_r = 9.9$ and a loss tangent $tan\,\delta = 0.003$. The parametric macromodel of the scattering matrix is built as function of the varying length of each folded segment $L \in [1.97, 2.41]$ mm and varying spacing between a folded stub and the main line $S \in [0.06, 0.24]$ mm over the frequency range $[5, 20]$ GHz. The EM simulation engine used is ADS Momentum.

In order to objectively assess the accuracy of the models, a dense $50 \times 30 \times 151$ $(L \times S \times frequency)$ reference grid was calculated. It is important to note that this dataset is not used during the modeling process in any way since typically such a reference grid is not available. It is simply used to objectively test the quality of the models a posteriori.

10.8.2 Experimental setup

The SUMO Toolbox is configured with the generic ANN and multivariate rational modeling plugins. The multivariate rational models are based on a custom implementation [155]. The order selection is performed using a genetic algorithm (population size: 30, number of generations: 20), thus this need not be done manually. The model accuracy estimator is 5-fold cross validation [204, 425] configured with a Mean Square Error (MSE) error function. The ANN models are based on the Matlab Neural Network Toolbox and are trained with Levenberg-Marquard backpropagation with Bayesian regularization [271,272] (600 epochs). Since the ANN models do not support complex data directly, the real and imaginary components are fitted separately using an ANN model with two outputs. The topology and initial weights are determined by an evolutionary strategy-like algorithm, with 25 models being generated each modeling iteration. To assess the model quality and drive the topology selection is taken as the sum of two criteria is optimized by the evolutionary strategy: the in-sample error (using a MSE) and the Linear Reference Model (LRM) score (chapter 9). The combined scores for each output (real/imaginary) are then added together to obtain the overall score of the model. The LRM score penalizes a model if it exhibits unwanted bumps or ‘ripples’ between the sample points. It can be seen as a kind of smoothness penalty that has the added benefit of keeping the neural network model complexity low. The advantage of using these two metrics together is that they produce better ANN models and are much faster to evaluate than cross validation.

The modeling starts with a Latin hypercube design of 4 points augmented with the corner points in the 2-dimensional $L \times S$ space. Each iteration a new sample is selected using the LOcal Linear Approximation-Voronoi (LOLA-Voronoi) adaptive sampling algorithm [239]. Because frequency is sampled automatically by ADS Momentum, LOLA-Voronoi samples in the 2-dimensional instance space defined by the geometric parameters L and S. New samples are submitted to ADS Momentum, which returns a set of S-parameters over the frequency range of interest. In order to select a sample in the reduced 2-dimensional design space (without the frequency parameter), slices are taken at multiple frequencies, and LOLA-Voronoi is used on each slice separately. This results in a nonlinearity estimation for each frequency slice, covering the entire

2-dimensional design space. These estimations are aggregated into one score, which is used to select new samples in locations with the highest nonlinearity over the entire frequency range.

Momentum is configured to return 31 frequency samples and the SUMO Toolbox is set to terminate after 136 instance simulations ($136 \times 31 = 4216$ data points). Thus instead of using an explicit target accuracy value, simulations are performed until the computational budget is exhausted.

10.8.3 Results

Figure 10.17 shows a plot of the final rational models for S_{11} and S_{12}. For space reasons the following discussion will only treat S_{11}. The results for S_{12} are completely analogous. A snapshot of the sample distribution after 14 Momentum simulations (434

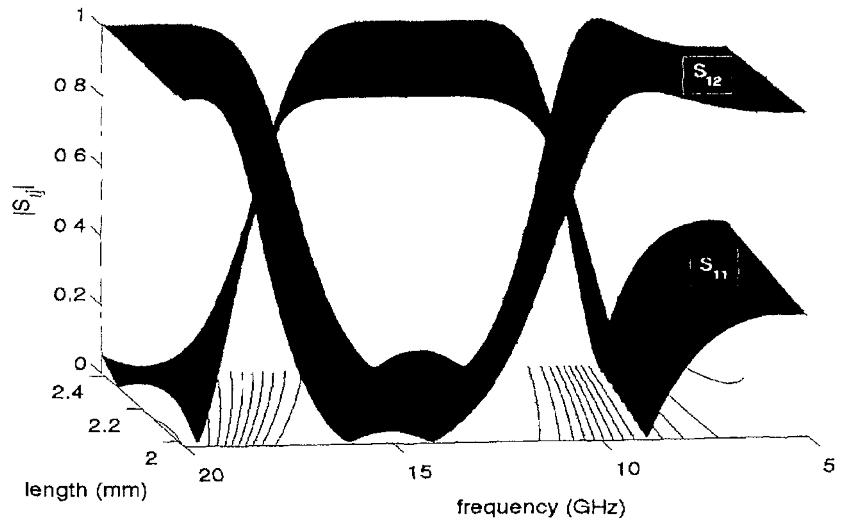


Figure 10.17: Plot of the final rational model for $|S_{11}|$ *and* $|S_{12}|$ *at* $S = 0.131$

points) is shown in figure 10.18. The figure nicely illustrates how the adaptive sampling only occurs on the geometric design parameters and that the frequency is densely sampled by Momentum itself. This is significantly more efficient that adaptively sampling in the full parameter space.

Figure 10.19 shows that the ANN model generation code is able to reduce the true error (evaluated over a dense reference grid) quite effectively while minimizing the estimated error. This means that the in-sample error/LRM combination is a good approximator of the true accuracy. In line with previous results [241] we see that the decrease in true error is quite steady with no large jumps. This can also be seen from figure 10.20 which shows how the *distribution* of the reference data errors evolve during the model generation process. The figures show a steady increase of the lighter areas which means an increasing proportion of the reference data points have low error.

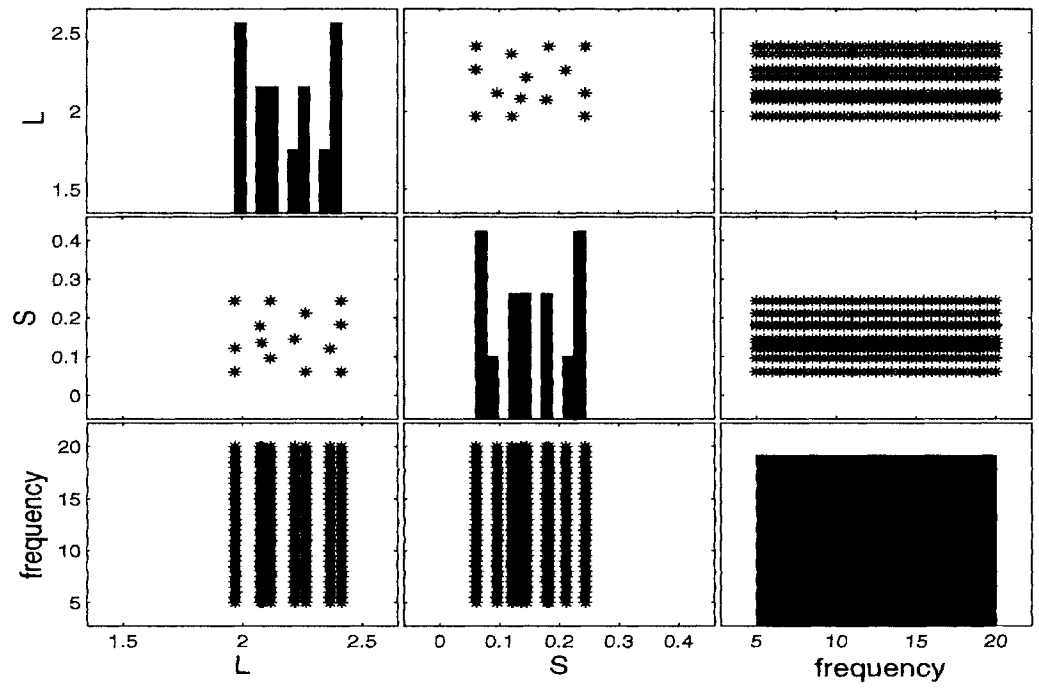


Figure 10.18: Snapshot of the sample distribution for S_{11} after 14 simulations (rational models). Plot (i, j) with $i \neq j$ shows the sample points projected onto the i^{th} and j^{th} dimensions. Plot (i, j) with $i = j$ shows the sample histogram for dimension i ($L = 1, S = 2, frequency = 3$).

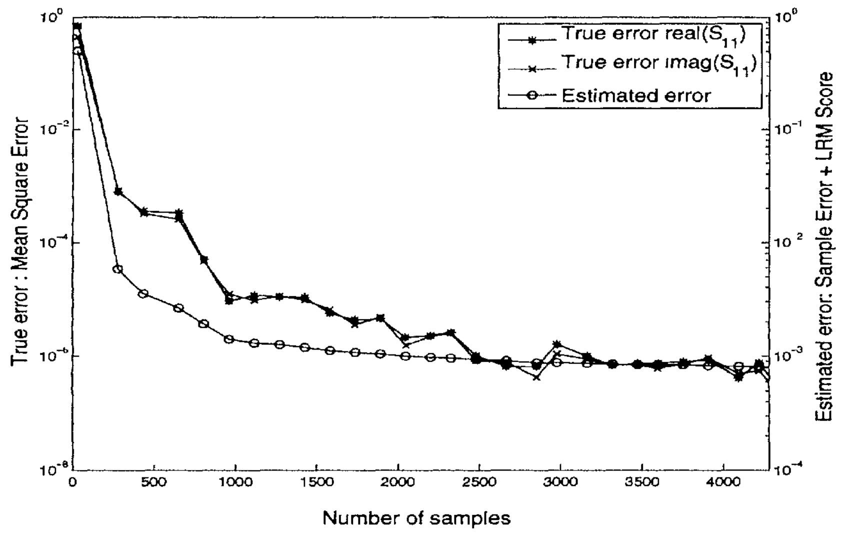


Figure 10.19: Evolution of the true and estimated error over the reference data during the ANN topology optimization. Note the true error is not minimized directly, it is only shown here for reference. The number of samples should be divided by 31 to obtain the number of Momentum simulations.

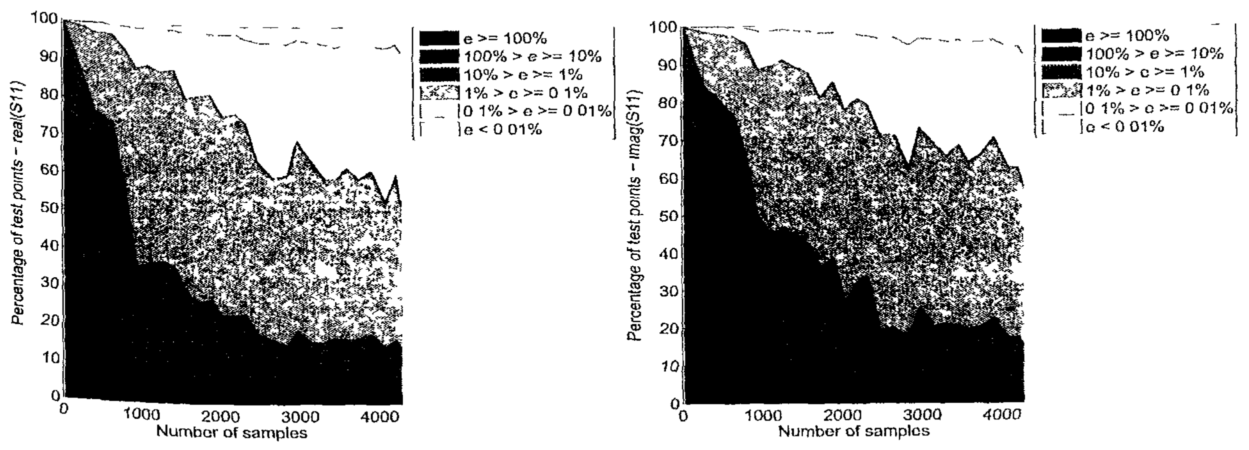


Figure 10.20: Evolution of the true error histogram (using a relative error) over the reference data during the ANN topology optimization. More lighter regions mean a higher percentage of the reference points have a low error. The number of samples should be divided by 31 to obtain the number of Momentum simulations.

The error histogram (using an absolute error) for the final best ANN model found by the toolbox after 136 Momentum simulations is shown in figure 10.21. The figure is obtained by using kernel density estimation [469] to estimate the true probability density function of the errors. The figure shows that very good accuracy is achieved. Note also the small difference between the training and test curves, meaning there is no overfitting and the models show good generalization. The final ANN model for S_{11} is a $3-8-16-2$ network (210 parameters) and for S_{12} a $3-12-15-2$ network (275 parameters).

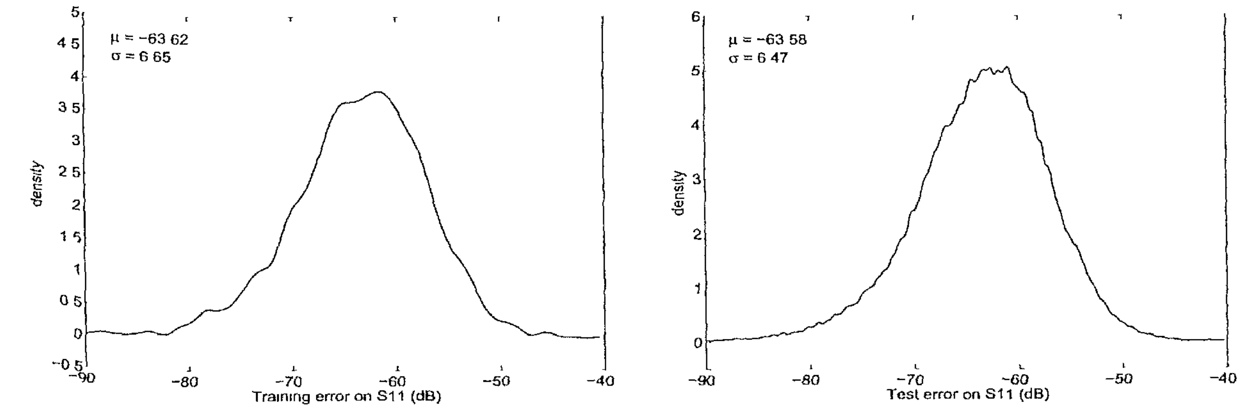


Figure 10.21: Probability density function of the absolute errors of the final ANN model over the training and reference data after 136 simulations.

Let us now regard the results for the multivariate rational functions. The evolution of the error histogram of the best model on the reference data during the model generation process is shown in figure 10.22. Compared to the ANN results in figure 10.20 we see that the error reduction is more erratic in the rational case. Particularly in the beginning, when only little data is available. This is due to the use of cross validation as the accuracy estimator. Even though we ensure an even distribution of the different folds, when data is relatively sparse cross validation is known to give biased results [425] and can mislead the order selection procedure. This can also be seen from figure 10.23.

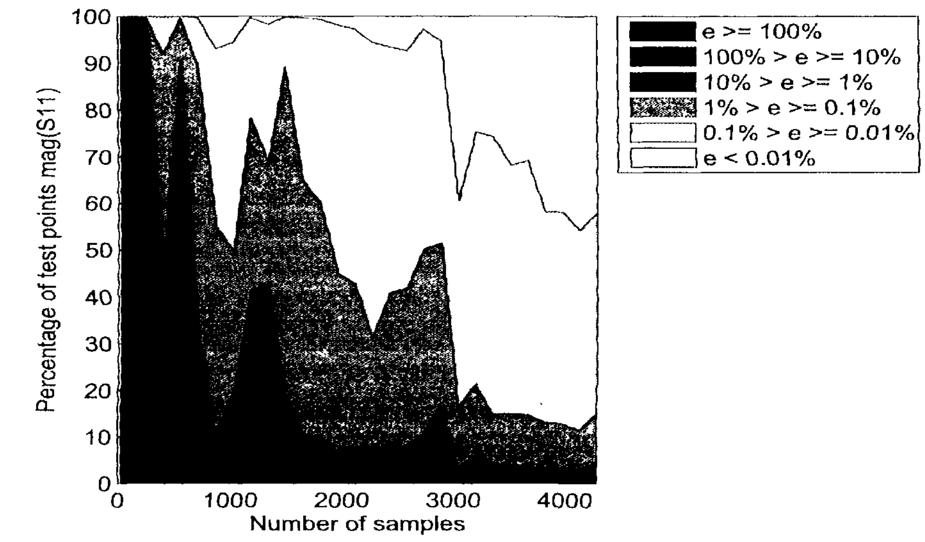


Figure 10.22: Evolution of the true error histogram (using a relative error) over the reference data during the rational model order selection. More lighter regions mean a higher percentage of the reference points have a low error. The number of samples should be divided by 31 to obtain the number of Momentum simulations.

The error histogram (using an absolute error) for the final best rational model found by the toolbox after 136 Momentum simulations is shown in figure 10.24. Again, good accuracy is achieved on S_{11}, with the mean accuracy being better than for the ANN models. Given the rational nature of the underlying transfer function this should not be a surprising. However, it should be noted that the modeling effort was not the same for both model types. Since the rational functions are fast to construct and train we can afford to build more of them during each modeling iteration than neural network models (since these are much slower to train). In this case $20\times30=600$ rational models are built each modeling iteration versus only 25 neural networks.

The final models used to plot figure 10.24 consisted of 513 parameters for S_{11} and 672 parameters for S_{12}. Note also the small difference between the training and reference data distributions, this means the models again show good generalization.

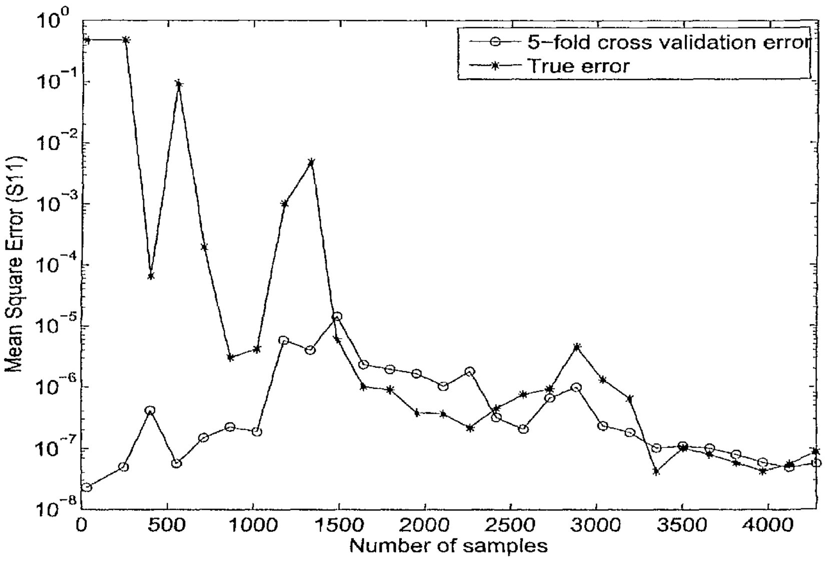


Figure 10.23: Evolution of the true error and estimated error over the reference data during the rational model order selection. The number of samples should be divided by 31 to obtain the number of Momentum simulations.

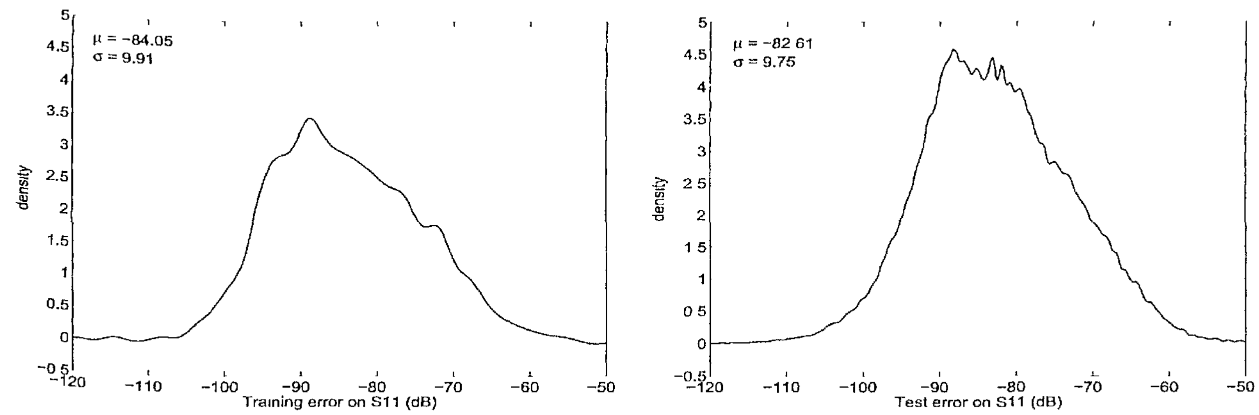


Figure 10.24: Probability density function of the absolute errors of the final Rational model over the training and reference data after 136 simulations.

10.9 Tapered Transmission Line

10.9.1 Background

We stay in the electronics domain. The next application involves a high-speed 2-port microwave structure. The goal is to model the complex electro-magnetic behavior of this system. This behavior can be characterized by the scattering parameters or S-parameters, which describe the response of an N-port system to signals at each port. These S-parameters are a function of the geometrical parameters of the system (length, substrate parameters, dielectric constants, ...) as well as the frequency. Like in the previous application, the frequency is again treated as a special parameter.

In this particular example we model the (complex) reflection coefficient S_{11} of a lossless exponential tapered transmission line terminated with a matched load, as described in [224, 470]. This structure is displayed in Fig. 10.25, where Z_0 and Z_L represent the reference impedance and the load impedance respectively. The reflection coefficient S_{11} is modeled as a function of the relative dielectric constant $\varepsilon_r \in [3,5]$ and the line length $L \in [1cm, 10cm]$, as well as the frequency $f \in [1kHz, 3GHz]$. The simulator, however, only accepts 2 inputs: ε_r and L. For each combination of these two inputs, the output for a complete set of frequency ranges, linearly distributed over the frequency range, is computed and returned for modeling.

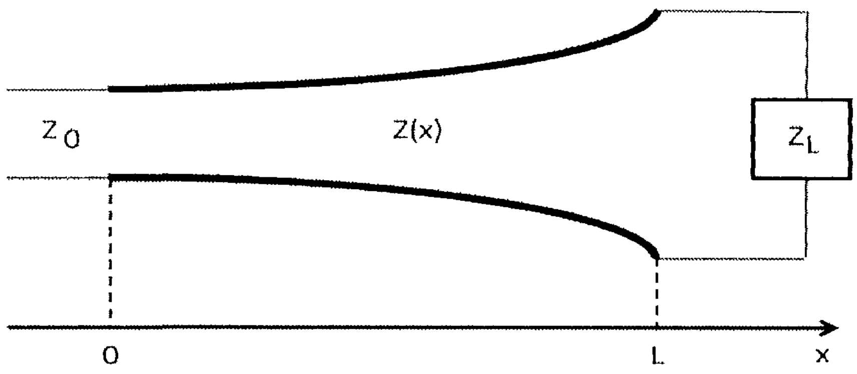


Figure 10.25: An exponential tapered micro strip transmission line. $Z_0 = 50\Omega$ *and* $Z_L = 100\Omega$.

10.9.2 Experimental Setup

This system was modeled with ANN models using a genetic algorithm to optimize the topology and the initial weights. The GA was run for 10 generations between each sampling iteration with a population size of 10. The network itself was trained using Levenberg-Marquardt backpropagation in conjunction with Bayesian regularization [272] for 300 epochs. The initial experimental design is a Latin hypercube design of 9 points in the 2-dimensional geometric parameter space. For each of these points, 30 frequencies values were returned, resulting in a total of 270 initial data points. From then on, each sampling iteration LOLA-Voronoi selects 2 additional sample locations, resulting in 60 more data points, up to a total of 1000.

Because the ANN implementation that is used does not support complex data directly, real and imaginary parts were modeled separately. Two different approaches were tried. In the first approach, a separate model was trained for each component of the complex output. In the second approach a single model was trained to model both components (as two separate real numbers), resulting in a considerable gain in speed at the (possible) cost of some accuracy. At the end, these two real outputs are then combined again to produce a model of the original complex output parameter.

10.9.3 Results

The results are summarized in Table 10.4. A plot of the modulus of the best model produced using separate neural networks for the real and imaginary part can be found in 10.26. The accuracy of the models was calculated by testing the models on a dense test set of 6750 points which was not used for training. Two measures were calculated: the root relative square error (RRSE) and the maximum absolute error (MAE).

	ANN Split			*ANN Combined*		
	Real	*Imaginary*	*Complex*	*Real*	*Imaginary*	*Complex*
RRSE	3.38E-3	3.19E-3	3.31E-3	4.60E-3	6.18E-3	5.19E-3
MAE	1.85E-4	1.42E-4	2.57E-4	3.00E-4	3.13E-4	4.73E-4

Table 10.4: The error of the models on a external test set of 6750 points. The results from training with separate networks are on the left, the results from training with only one neural network are on the right (tapered transmission line example).

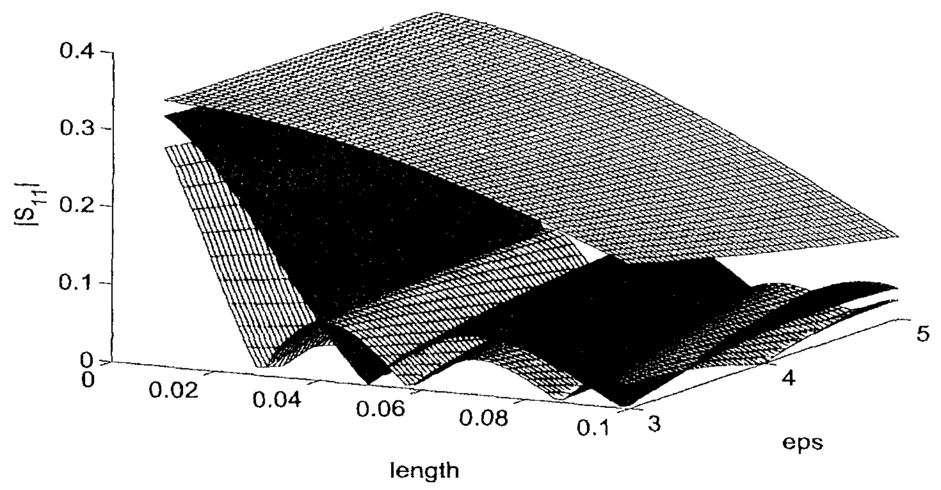
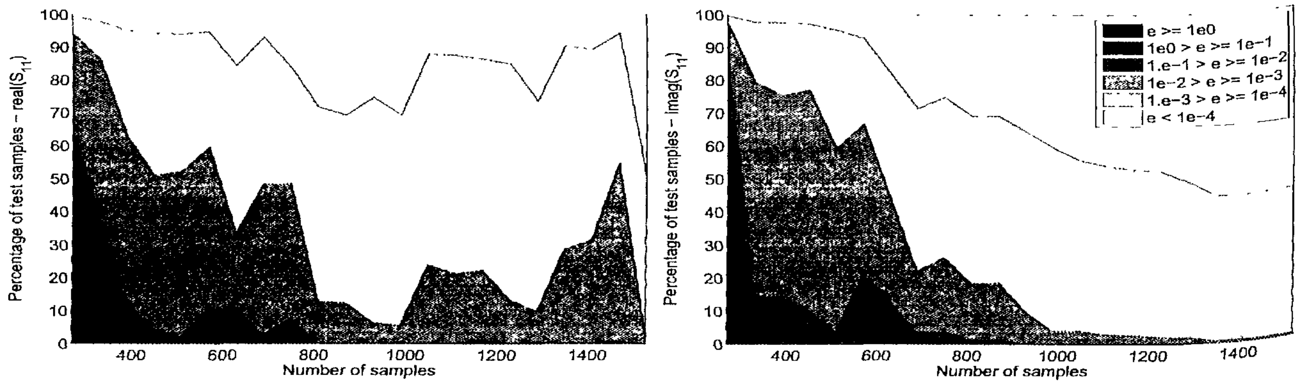


Figure 10.26: A plot of $|S_{11}|$ using two separate networks for the real and imaginary components (tapered transmission line example). The model is shown at three frequency slices (minimum, maximum, and middle frequencies).

As expected, by training the real and imaginary part with a separate neural network, a small accuracy improvement can be obtained at the cost of a longer run time. However, both approaches performed very well, producing models with a root relative square error of less than 0.01 after only 570 samples. By adding another 1000 samples, the accuracy can be further improved by almost a factor 10. This can be seen in Fig.

10.27. The left plot shows the evolution of the error on the real part, while the right plot shows the evolution of the imaginary part. This plot represents the percentage of samples in the test set that have an error in the a range corresponding to a particular shade of grey on the plot.



(a) real(S_{11}) (b) imag(S_{11})

Figure 10.27: Evolution of the test error histogram using a relative error (tapered transmission line example).

This plot shows that the error decreases steadily as the number of samples increases for the imaginary part, but oscillates slightly for the real part. When more samples are evaluated, the model might have to find new optima for its hyperparameters, thus temporarily causing a drop in accuracy while the parameter space is explored.

10.10 Others

Finally, we give a couple of uncategorized examples of data that has been fitted with the SUMO Toolbox. As mentioned at the start of this chapter, more supporting material or movies can be found at `http://sumolab.blogspot.com` and `http://www.youtube.com/user/sumolab`.

Figure 10.28 shows a model of the 2D Schwefel function, its mathematical formulation being

$$f(\mathbf{x}) = 418.9829 \cdot d + \sum_{i=1}^{d} x_i \sin\left(\sqrt{|x_i|}\right) \tag{10.8}$$

with x_i defined on $[-500, 500]$.

Much less regular and smooth is the Langermann function, defined as

$$f(\mathbf{x}) = -\sum_{i=1}^{m} c_i \left(e^{\frac{||\overline{\mathbf{x}}-A(i)||^2}{\pi}} \cdot \cos\left(\pi \cdot ||\overline{\mathbf{x}} - A(i)||^2\right)\right) \tag{10.9}$$

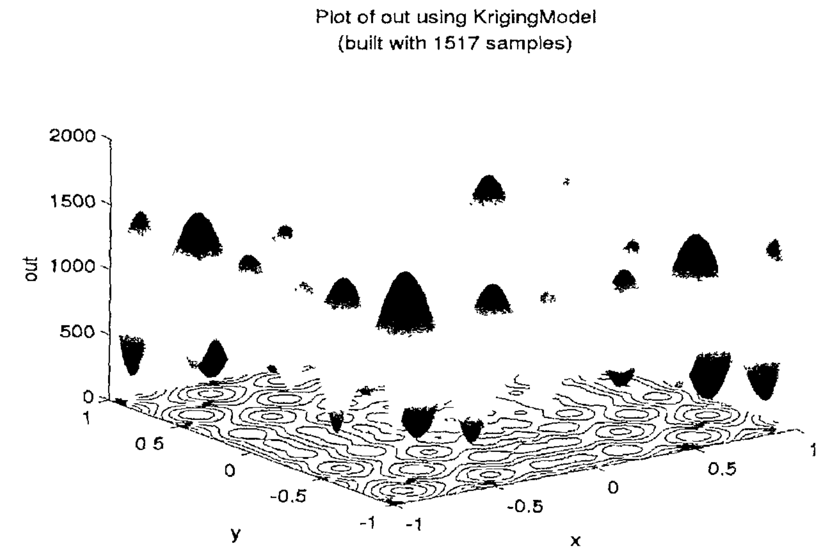


Figure 10.28: Kriging model of the Schwefel function after 1517 points.

with $m = 5$ and $x_i \in [0, 10]$. The values of the matrices A and c can be found in the implementation. An LS-SVM model of this function is shown in figure 10.29.

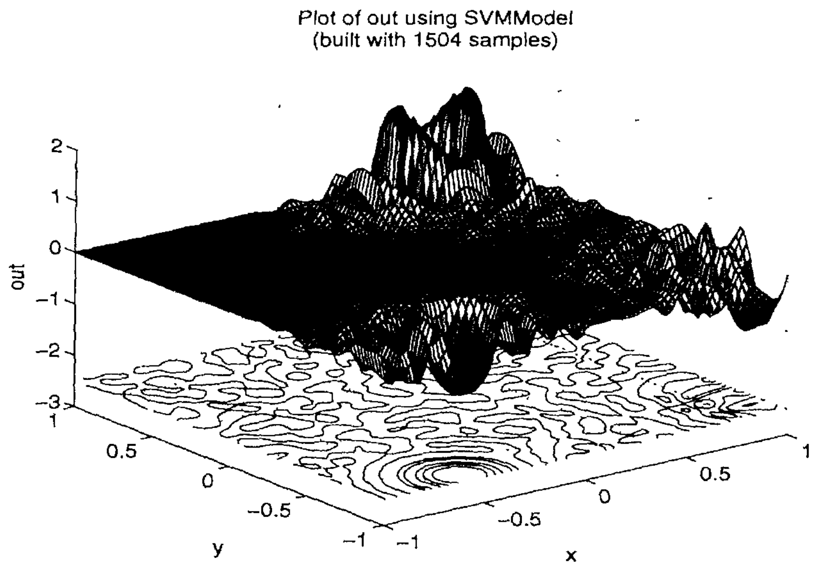


Figure 10.29: LS-SVM model of the Langermann function after 1504 data points.

Figure 10.30 then shows a Kriging model of a completely different type of data: geometric data resulting from a high resolution laser scan of Michelangelo's David.

In section 4.9 we discussed how the SUMO Toolbox framework can also be used to solve classification problems. To illustrate this we applied SUMO to the classical two-sprial problem where the goal is to separate two intertwined spirals. Using the

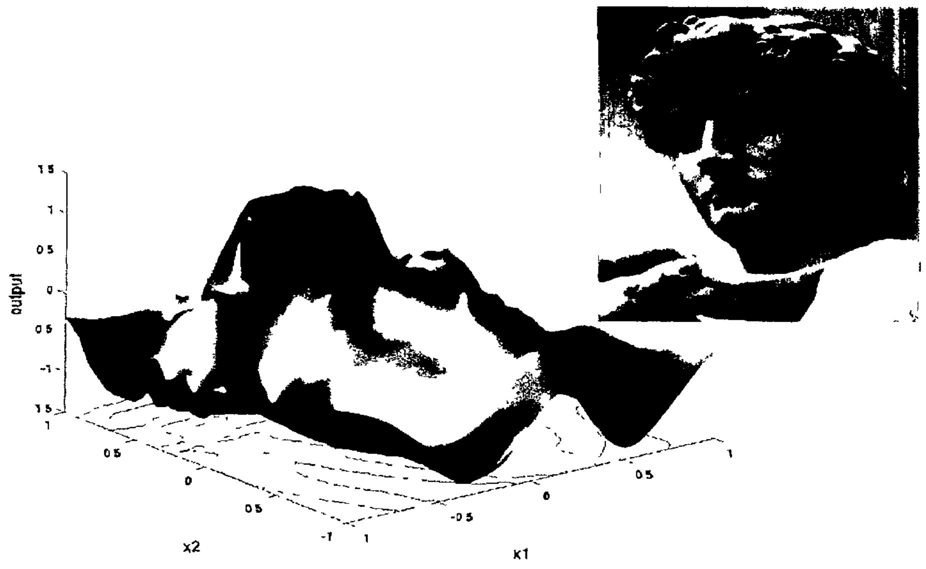


Figure 10.30: Kriging model of geometric data of the face of Michelangelo's David. Data courtesy of the Digital Michelangelo Project from Stanford University.

SVM plugin with adaptive sampling and hyperparameter optimization this can easily be done. A plot of the final classifier is shown in figure 10.31.

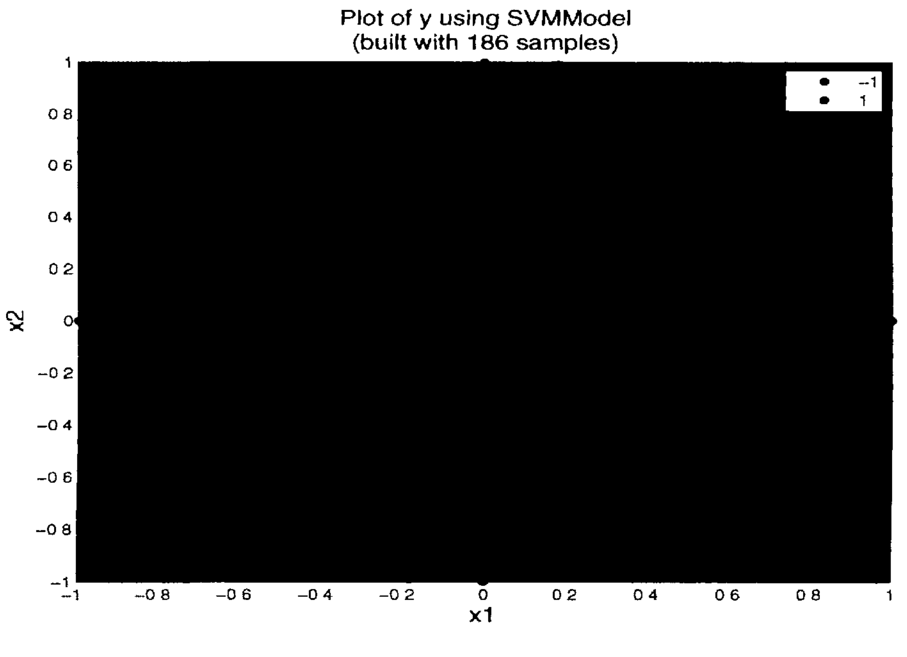


Figure 10.31: Final SVM classifier solving the two spiral problem.

An interesting benefit of a classification approach (though a regression approach would work as well) is that this can be applied to the modeling of geometric data from arbitrary three dimensional objects. The idea is to create an analytic model to represent a 3D structure. Popular techniques for this are RBF models (the FastRBF library) or Neural Models (spherical Self-Organizing-Map, neural-GAS). For the purpose of this dissertation we just give a proof of principle example to show how the SUMO Toolbox infrastructure could be used to achieve this using the standard plugins.

The approach is simply to regard the 3D geometric modeling problem as a classification problem where the function value of a 3D Cartesian point maps to -1,+1, or 0 if the point is situated inside, outside, or on the object. If the object is represented as a triangular mesh this can easily be calculated, though it requires the object to be closed. In essence this is a 3D classification problem which can be solved just as any other. Once the model is fitted on the data one simply needs to plot the isosurface of the model at zero in order to obtain the object. Figure 10.32 shows a simple example for a 3D model of a cactus. The figure shows the isosurface of an SVM classifier trained on the geometric data. The training was done on relatively few points so the surface is not very smooth. Using more data and putting more effort into the modeling process will improve the results, but we simply show it here as a proof of concept illustration.

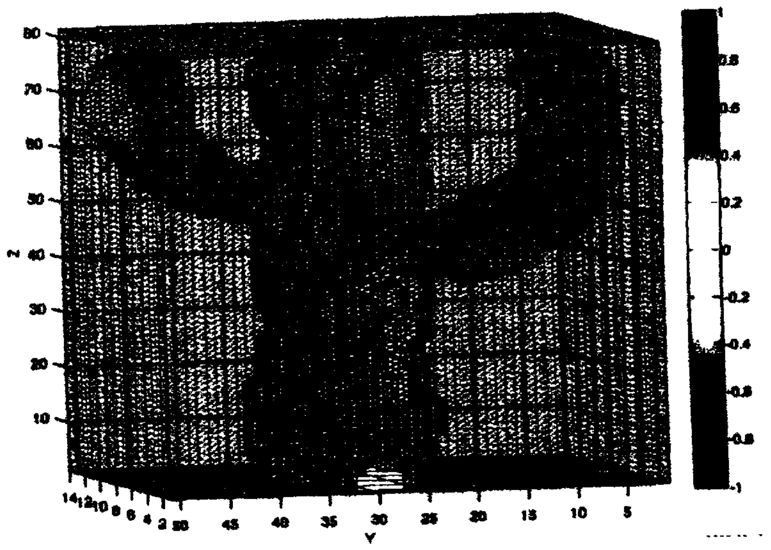


Figure 10.32: Proof of concept illustration showing the zero-isosurface of an SVM classifier trained on 3D geometric data from a cactus. Data courtesy of www.advancedmcode.org.

11 Conclusion

When you have completed 95 percent of your journey, you are only halfway there.

– Japanese proverb

11.1 Summary

In many science, and engineering problems researchers make heavy use of computer simulation codes in order to replace expensive physical experiments and improve the quality and performance of engineered products and devices. Such simulation activities are collectively referred to as computational science and computational engineering.

Unfortunately, while allowing scientists more flexibility to study phenomena under controlled conditions, computer simulations require a substantial investment of computation time. One simulation may take many minutes, hours, days or even weeks, quickly rendering parameter studies impractical. Of the many ways to deal with this problem, this dissertation has focused on the use of data based global surrogate models for a particular class of problems (cfr. section 1.1.2). The goal being to generate a surrogate that is as accurate as possible, using as few simulations as possible, and with as little overhead as possible. Once such a surrogate is available, it can be reused further down the engineering design pipeline.

In the course of this dissertation we have discussed the motivation that drives surrogate modeling research (chapter 2), described in depth the different sub problems that are involved in applying it successfully (chapter 3), and a comprehensive software platform that brings together different approaches in a flexible, re-usable, extensible framework (chapter 4). The importance of the availability of ready made software is worth emphasizing. Without backing in software many ideas and new algorithms remain confined to the papers that describe them, using a format and applications that are often hard or impossible to reproduce independently. The availability of a toolbox implementing all algorithms discussed in this thesis at least allows for full individual experimentation, comparison, and augmentation.

Besides these topics, this dissertation also explored how concepts from distributed computing (chapter 6), evolutionary computing (chapter 7), and multi-objective optimization (chapter 8) can be used to further improve the surrogate modeling process. Using a large number of problems from many different application domains as illustrating examples. Thus, in summary, this thesis has operated on the meta level. It has not focused on one particular technique or problem, but rather on how existing methods and technologies can be combined and integrated in an efficient way in order to optimally benefit a domain expert and the solution of his design problem.

11.2 Research challenges

Reflecting back on the research challenges listed in section 1.2, the overarching challenge was to fill in the question mark in figure 11.1 with a workflow that can produce an accurate global surrogate model at a minimum computational cost and overhead.

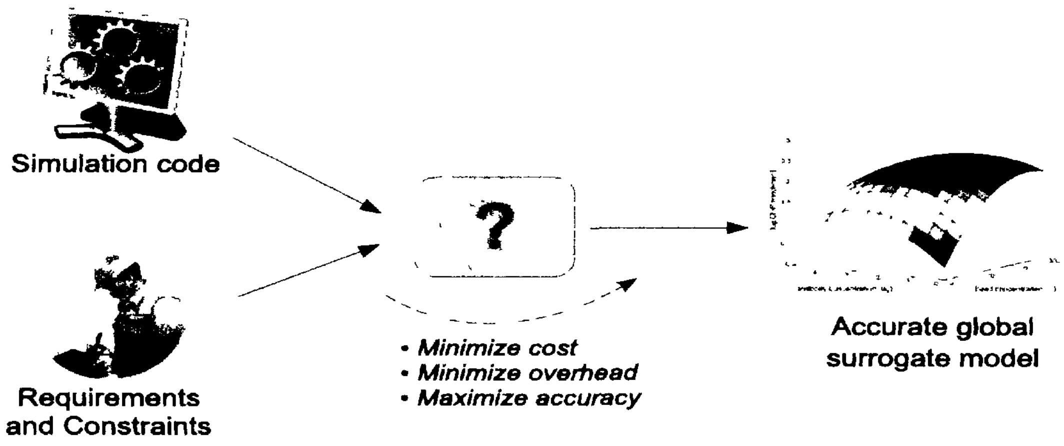


Figure 11.1: Surrogate modeling research challenge

This dissertation has tackled this challenge through the design and implementation of the SUMO Toolbox framework which is discussed in depth in chapter 4. Given a simulation engine or other data source that meets the requirements from section 1.1.2, the SUMO Toolbox can adaptively generate an accurate global surrogate model within the accuracy and budget requirements defined by the user. Judging by the many different uses of the SUMO framework (section 11.3), the many downloads (section B.4), and the authors' own experience from interacting with users; we can conclude that the SUMO framework is quite successful in replacing the question mark in figure 1.2. Thus, bearing in mind the critical remarks from section 4.10, the main research challenge has been accomplished.

Concerning the generality-specificity sub-challenge, as discussed in the dissertation, theoretical work from machine learning [164] suggests this is impossible to solve in an a priori manner (see also section 3.7.2). Many advanced algorithms have been presented for model parameter optimization [21, 177, 358], adaptive sampling [157, 471], model selection [127, 170, 472], and knowledge inspired modeling [74]. However, no algorithm is optimal in all circumstances [4, 44, 67]. The optimal algorithm will depend on the problem characteristics and is usually very hard to determine up front. To add to the difficulty, there is a complex dependency web between the different sub-problems. The best sampling strategy will depend on the model type, which is linked with the hyperparameter optimization strategy, which in turn, depends on the model selection method, etc.

All the more reason that it should be very easy to try out, add, and compare different algorithms and approaches, a core design decision of the SUMO framework. For while one can never guarantee the optimal solution, a flexible system that allows different approaches to be easily tried, possibly with some added automation, will still lead to good solutions which satisfy the requirements. While at the same time giving more confidence that the different alternatives have been explored in a systematic manner.

This brings us to the accessibility sub-challenge, ensuring that advanced surrogate modeling algorithms can easily be used and integrated into the larger design process. This has been achieved on different levels (see also sections 4.7 and 4.8):

- *Simulator level*: the requirements on the format of the data entering the SUMO framework have been kept as simple as possible, i.e., ASCII based, one line per point. A GUI is currently under construction to also automate the generation of the necessary XML files. In addition the different *SampleEvaluator* classes make it straightforward to connect any data source (text file, shell script, executable) with the framework and new subclasses can be added to support more complex data formats (e.g., Microsoft Excel files).

- *Framework level*: by adopting a modular design and minimizing the complexity of the plugin API, the code is logically structured and can be reviewed or changed if needed. New plugins can easily be added if the basics of object oriented programming are understood and a large number of plugins are already available that can serve as guiding examples. In addition, the whole framework is available under an open source license (The Affero GNU General Public License[1]) thus encouraging code contribution and reuse.

[1]`http://www.fsf.org/licensing/licenses/agpl-3.0.html`

- *Application level*: once a model has been generated, a user friendly GUI tool is available to allow a domain expert to explore the model in an intuitive fashion. Many helper methods are also available that provide access to information like derivatives and prediction uncertainty estimation. The model objects can be compiled to standalone C code using the Matlab to C compiler or can be exported to a standalone Matlab script or mathematical expression. While this export functionality is currently not available for all model types, it is just a matter of implementation. A further improvement would be to also integrate export functionality to standard commercial CAD/CAE tools like ADS Momentum or Simulink.

In addition some work is currently underway in order to expose SUMO Toolbox instances on the network (cfr. section 6.6), facilitating integration even further.

11.3 The SUMO Toolbox: Applications and Users

Since the SUMO framework formed the cornerstone of this dissertation, it is worth reviewing the applications the framework has been involved in. The application domains covered in this dissertation include:

- **Electronics**: chapter 5, sections 10.7, 10.9, and 10.8
- **Hydrology**: section 10.3
- **Structural dynamics**: section 9.4.2
- **Chemistry**: section 10.4
- **Automotive**: section 8.5.3
- **Metallurgy**: section 10.6
- **Demography**: section 7.8.5
- **Aerodynamics**: sections 8.5.4 and 10.5
- **Bio-physics**: section 6.7.2

Other applications not mentioned in the dissertation and which are currently ongoing, have recently been published, or will be published in the near future are:

- Material Characterization of patch antennas, EM Group, Dept. of Information Technology, Ghent University
- Modeling of the electrical behavior of a H-Antenna, IBCN Group, Ghent University

- Magnetic Material Characterization, Lab on Electrical Energy, Systems & Automation, Ghent University [473]
- Geometric design optimization of a microwave narrow-band filter, SUMO Lab, Ghent University [474]
- Trade-off analysis of LNA performance parameters, NXP Semiconductors, The Netherlands

The above examples are applications where one of the research group members is closely involved. The SUMO Toolbox has seen quite a few uses and applications by third party researchers. Of those applications, the ones that we are aware of that are currently ongoing include:

- Flocculant adsorption, CSIRO, Australia [475]
- Electronic Packaging, North Carolina State University, Raleigh, USA [476]
- Modeling the chemical processes in fuel cells, Fuel Cell Materials and Manufacturing Laboratory, University of Toronto, Canada
- Modeling and optimization of the Gas-Metal-Arc welding process, Welding and Joining Institute, RWTH Aachen, Germany
- Multi-class classification and data mining, Kaunas University of Technology, Lithuania

For more information regarding downloads and usage refer to appendix B.

11.4 Future Work

The path of scientific research is of course one that is continuously ongoing. This dissertation has covered a broad range of topics and applications, and much is still left to be explored. Some of these topics of future work have already been covered in section 4.10.

Most pertinent in this area is the issue of parameter screening or dimensionality reduction. The number of variables that can be routinely tackled is continuing to rise. There is a constant demand from domain experts to be able to take into account more and more parameters. Due to the curse of dimensionality global models quickly become infeasible and current data fitting methods can no longer be applied with high accuracy. However, as Keane and Nair state in [4], one should keep in mind that:

> *Across domains like vision, speech, motor control, climate patterns, human gene distributions, and a range of other physical and biological sciences, various researchers have reported evidence that corroborate the*

fact that the true intrinsic dimensionality of high dimensional data is often very low [..]. We interpret these findings as evidence that the physical world has a significant amount of coherent structure that expresses itself in terms of strong correlations between different variables that describe the state.

Focus must be placed on dimensionality reduction methods, ranging from statistical variable screening methods to domain decomposition and local learning approaches. The challenge is to ensure such techniques can readily and easily be applied by non-experts through readily available software. Integration of such a framework for parameter screening into the SUMO Toolbox would greatly increase its usefulness and appeal. Specialized support for discrete parameters would also help in this respect.

A second challenge linked with the problem of increased dimensionality is the integration of domain specific knowledge. Besides dimensionality reduction this is the only way to ensure problems of the future can be tackled effectively. Again much research on knowledge integration techniques has been conducted. The challenge in the future is to to make them easily accessible to non-modeling experts so they can easily be applied to existing data fitting types and application domains. The addition of such a framework to the SUMO Toolbox would again be a good asset.

The difficulty of such works is to sufficiently hide the complexity of the modeling problem from the user without compromising performance. Adaptive algorithms with self-regulating parameters are a powerful and promising way to achieve this. However, dealing with the computational cost they incur is non-trivial. This is an area that will definitely benefit from further hardware developments.

The author also sees a promising application of the ideas from fuzzy theory and knowledge capturing techniques to the surrogate modeling process. This would allow a domain expert or engineer to express his modeling requirements in a more natural way. Progress in natural language processing will also help in that respect. Convergence of these fields, though far from trivial, holds promising advances in design and optimization software and interaction models.

A

Evolutionary Neuro-Space Mapping Technique for Modeling of Nonlinear Microwave Devices

A.1 Introduction

Modeling and computer-aided design (CAD) techniques are important in helping microwave designers to achieve efficient design of linear and nonlinear microwave circuits. In recent years, artificial neural networks (ANNs) [477] and space mapping [76] have been recognized as two important developments in microwave CAD to address the growing computational challenges in modeling, simulation and optimization. ANNs can be trained to learn microwave component data, and the trained ANNs can be used as fast and accurate models for efficient high-level circuit and system design [20, 74, 478–481]. On the other hand, space mapping is an advanced optimization concept, successfully used to achieve substantial computational speedup in otherwise expensive optimizations of microwave components and circuits [229, 482–484]. Techniques combining ANNs and space mapping have also been developed for electromagnetic modeling [478], nonlinear device modeling [230] and statistical device modeling [485].

This paper explores further advances in the application of ANN and space mapping for modeling of nonlinear microwave devices. Nonlinear device modeling is an important area of microwave CAD, and many device models have been developed [486, 487], such as physics-based models, e.g., [488–490], equivalent circuit based models, e.g., [491–496] or table based models, e.g., [497]. With the continuous development of semiconductor device technologies, new devices constantly evolve. Models that were developed to fit previous devices may not fit new devices well. There is an ongoing need for new models. The need for faster model development cycle also demands new CAD methods for modeling, so the task of model development becomes more efficient and systematic.

Recently, a CAD method for nonlinear device modeling, called Neuro-Space mapping (Neuro-SM) technique has been introduced [230, 498]. It is a systematic computational method to address the situation where an existing device model cannot fit new device data well. The methods start from a known equivalent circuit model that is already a coarse approximation of the new device behavior. We refer to this existing equivalent circuit model as the coarse model. A generic method like ANN is then applied to map or modify the voltage/current relationship in the coarse model to match that of the new device data. The final model, i.e., the Neuro-SM model, is a combination of both the ANN mapping and the coarse model. The ANN part of the overall model is referred to as the mapping structure, and the existing equivalent circuit model as the knowledge that is integrated with the ANN. In this way we can "repair" (i.e., improve) an existing model by a CAD method such that the final model matches the new device much better. The specific mapping structure for nonlinear device modeling introduced in [230, 498] was based on an input-mapping concept, which is a type, and the earliest type of space mapping originally formulated for EM optimization [229]. There is no guarantee that the input-mapping structure is the best mapping structure for all device examples. For example, the input mapping may not be sufficient if the output of the fine model is beyond the output range of the coarse model. Fortunately, a variety of mapping methods have been developed in passive/EM modeling that can be adopted for nonlinear device modeling, such as space mapping [76, 230, 483–485], difference mapping (or difference method) [74], output mapping [228], prior knowledge input (PKI) method [228], and knowledge-based neural network involving hybrid mapping [74, 499].

The overall efficiency of a general Neuro-space mapping model depends on the quality of the equivalent circuit model and the suitability of the mapping structure. In order to obtain optimal overall model accuracy it is important that the best combination of equivalent circuit model and mapping structure is used. Which model/mapping combination performs best is difficult to determine up-front since it depends on the data and the problem characteristics [483, 499]. Thus, this problem must be solved on a case-by-case basis, each time a new device is considered. However, there are many combinations of possible equivalent circuit models and mapping structures. It is very expensive to fully exploit this rich combination due to the cumbersome manual process of selecting structural combinations and optimizing the model and mapping parameters. This also means that there is still a large potential for improvements in accuracy if the space of different combinations is searched more efficiently. This brings us to the motivation of this paper.

We present a new methodology for nonlinear device modeling that is not tied to a particular device, an empirical approximation, or a mapping structure. The paper describes an approach to explore the space of mapping structures by formulating the Neuro-SM structural optimization problem with an evolutionary optimization algorithm. To enrich optimization search space and extract maximum possible improvements in modeling with simpliest mapping function, we also break down the mapping structure into finer structural variables for optimization. For example, separate mapping structures for voltage mapping, current mapping, transistor gate mapping, or drain mapping. In addition, since the efficiency of the mapping depends on the quality of the nonlinear equivalent circuit model we go one level deeper by decomposing the equivalent circuit models into their constituent internal branches and allow mixing of these branches across different model types. This further enriches the optimization search space, i.e., the different combinations of mapping structures and equivalent circuit models, allowing potentially superior models compared to traditional pure equivalent circuit models or hybrid models with predetermined mapping structures such as [230]. This also allows for simpler mapping functions which is desirable. In our implementation of the optimization, the final solution (a model) from the optimization program is a simple netlist representing the complete model structure, and values of parameters of all the functions in the model.

A.2 Evolutionary Knowledge-Based Modeling

A.2.1 Motivation

The starting point of our work is that the existing model cannot match the new device behavior. The existing equivalent circuit models, called coarse models, need to be modified and extended in order to accommodate for new device behavior. Manual modification of models is a trial and error process and hybrid methods have been developed to help map the coarse model to the device data [76,228,230,483–485,499,500].

However, there are a variety of mapping methods and the optimal mapping choice is typically not well defined since it depends mutually on the problem and the nonlinear equivalent circuit model (the better the nonlinear equivalent circuit model, the simpler the mapping structure). This mutual dependence cannot be solved in an a priori manner since there is no mathematical procedure that can be used as a guide. Nevertheless, for optimal accuracy of the overall model it is important that both the mapping and the equivalent circuit model are optimally complementary.

The result is that the different combinations of mapping structures and nonlinear equivalent circuit models must be tried manually which is a cumbersome undertaking. This is even more true if one goes beyond homogeneous models but also optimizes the internal structure of the equivalent circuit models themselves (e.g., hybrids of different equivalent circuits). The advantage of this is that it allows the mapping itself to be simpler.

The result is a combinatorial explosion due to the cross-product of different mapping methods and structural variations of each equivalent circuit model type. Navigating this search space is very human intensive and can only be done for a limited set of possible solutions. Significant cost savings and potentially large gains in accuracy could be obtained if this search is automated and parallelized in an efficient way.

A.2.2 Genetic Algorithms

We propose the use of evolutionary search to tackle the aforementioned problem. The evolutionary paradigm is well suited for optimization in large discrete search spaces and has already been applied successfully in many circuit generation problems [501, 502]. In the context of this paper, an evolutionary algorithm starts from an initial population of mapping structures and nonlinear equivalent circuits (which may be randomly generated) and uses specific reproduction operators in order to generate new circuit models based on the previous population. Proper definition of the reproduction operators allows the algorithm to efficiently explore the large search space to quickly come to an optimal solution.

The genetic algorithm is probably the most well known evolutionary algorithm and is used as an algorithm for global search, optimization being the most widely used application. The core algorithm, as introduced by Holland [314] is referred to as the Canonical Genetic Algorithm and is presented in pseudo code in the Appendix. Genetic algorithms have found widespread use in many domains with applications in transportation [319], electronics [502], vehicle design [321], scheduling [322], data fitting [323], and many others. In particular, driven by the work by Koza et. al. [503], the new field of evolvable hardware [501, 504, 505] has led to some very innovative applications of the evolutionary paradigm to electronic circuit design problems.

An important advantage of evolutionary methods over classic optimizers such as Broyden–Fletcher–Goldfarb–Shanno or Simplex search [506] is that evolutionary methods are well suited to structural topology optimization while classic methods are more suitable to continuous optimization problems in a real valued space. An advantage over other direct search methods like Particle Swarm Optimization [507] is that the user can exercise fine control over the concrete representation and genetic operators used. This allows the genetic algorithm to be fine-tuned to the problem at hand, making the incorporation of problem specific knowledge, constraints, and requirements more convenient.

A.2.3 Chromosomal Encoding of the Model Search Space

The purpose of the genetic algorithm is to find the optimal mapping structure for an existing knowledge model. The associated topology of the equivalent circuit model, or knowledge model, can also be optimized automatically to explore the search space in order to come to an improved overall model.

In order to make this possible we must first formulate a coding scheme to represent the different mappings and associated equivalent circuits. This coding scheme must be a function that uniquely translates a mapping structure/equivalent circuit model combination into a code the genetic algorithm can work with. While traditionally a binary encoding was employed, ample research has shown that a real valued, more problem specific encoding can greatly increase search efficiency [317]. For this paper we take the building blocks of the code to be the different mapping types and the different elements of one or more existing equivalent circuit models.

In the next subsection we first discuss the base equivalent circuit model employed in this paper and how the different types of mapping structures are implemented around it. The subsequent subsection will then discuss how the equivalent circuit model itself can be encoded and its topology optimized.

A.2.3.1 Mapping Structure Implementation

This paper is concerned with modeling a transistor and we focus on the most difficult part of the transistor modeling, the nonlinear intrinsic part [508]. Assuming that the extrinsic elements are taken fixed and can be separately determined from S-parameter measurements [508], we only consider modeling the intrinsic circuit. We use a base circuit model with three nodes for the intrinsic modeling of the transistor device, i.e., the gate, the drain, and the source terminals, denoted by G, D, and S, respectively. We denote the branches connecting the three nodes by $B = \{GS, GD, DS\}$, with each branch containing one or more elements (e.g., diode, nonlinear capacitor, nonlinear controlled source, etc.). An example of such a circuit structure is shown in Fig. A.1. This is an example of the coarse model part in the overall model.

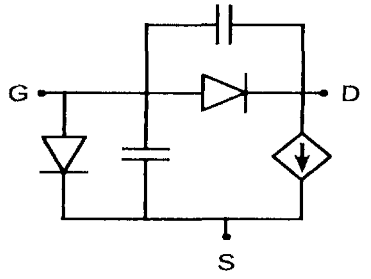


Figure A.1: Base circuit model example (the intrinsic circuit of a transistor) with three nodes (Gate, Drain, and Source). All branches, including the capacitors, are nonlinear elements in general.

We now temporarily assume a fixed topology for the equivalent circuit model and discuss the different mapping structures and how they are implemented. With mapping we mean that the gate and drain voltage signals of the device will not be applied directly. Instead, they will be modified (mapped) and only then applied to the coarse model. Similarly, the gate and drain current signals can also be mapped. However, there are many different ways in which the behavior of the coarse model may be misaligned with the true device data: Ranging from a simple misalignment easily corrected by a linear shift in the input space, to a highly nonlinear misalignment requiring a complex correction operator in the output space. Depending on the nature of the misalignment (simple, complex, in the input space or output space, etc.) a different mapping method, or combination of methods, will be required and it is not always obvious which mapping method is most suited.

Fig. A.2 shows the base circuit model example from Fig. A.1 extended with elements that make the different mappings possible. A common source implementation is considered here. The interpretation is as follows: given the gate and drain terminal voltage signals v_g and v_d as inputs, the gate and drain current signals i_g and i_d of the modeling problem are solved from those of the equivalent circuit model (defined as $v_{g_{map}}$, $v_{d_{map}}$, $i_{g_{map}}$, and $i_{d_{map}}$) through the application of input mapping, output mapping, difference mapping, or any subset thereof. We now discuss each of these three possibilities in turn.

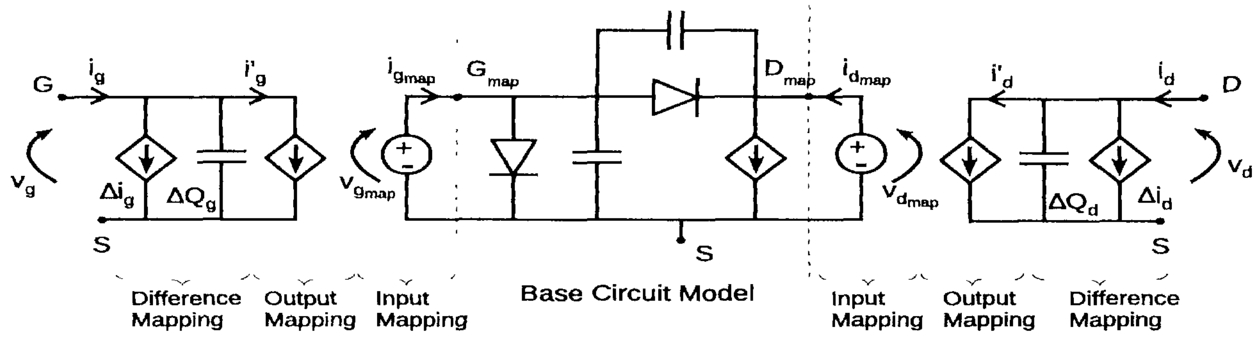


Figure A.2: Base circuit model from Fig. A.1 extended with mappings (represented by controlled sources in the circuit form). All three mappings (input mapping, output mapping, and difference mapping) are shown on the figure, both on the gate terminal and the drain terminal.

Input Mapping In the input mapping method, the mapping function maps the input space of the original problem onto a coarse model input space [230]. The coarse model is an existing equivalent circuit model that cannot represent a new device behavior accurately in the original input space. By applying input mapping, the coarse model with mapped inputs can produce the outputs with improved accuracy [230]. Input space mapping is achieved by adding voltage controlled voltage sources that perform a mapping of v_g and v_d onto $v_{g_{map}}$ and $v_{d_{map}}$ through

$$v_{g_{map}} = f_{IM_g}(v_g, v_d) \tag{A.1}$$

$$v_{d_{map}} = f_{IM_d}(v_g, v_d) \tag{A.2}$$

where $f_{IM_g}(\cdot)$ and $f_{IM_d}(\cdot)$ represent the input mapping equations for the gate and drain voltage signals, respectively. Input space mapping is most effective when only the input of the coarse model needs to be realigned [499]. The simpler this realignment (i.e., depending on the coarse model) the more straightforward the implementations of $f_{IM_g}(\cdot)$ and $f_{IM_d}(\cdot)$. Thus, a poor choice for the coarse model will make constructing an accurate mapping function considerably more difficult. If no input mapping is requested by the algorithm, an identity function is used, i.e., $f_{IM_g}(v_g, v_d) = v_g$ and $f_{IM_d}(v_g, v_d) = v_d$.

Output Mapping With output mapping, the appoximator learns the relationship between the outputs of a prior knowledge model and the original problem. We use the Prior Knowledge Input (PKI) formulation [228] of output mapping. Output mapping is applied on the output currents and input voltages of the equivalent circuit model using controlled current sources:

$$i'_g = f_{OM_g}(v_g, v_d, i_{g_{map}}) \tag{A.3}$$

$$i'_d = f_{OM_d}(v_g, v_d, i_{d_{map}}) \tag{A.4}$$

where $f_{OM_g}(\cdot)$ and $f_{OM_d}(\cdot)$ represent the output mapping equations for the gate and drain current signals, respectively. Hence, as opposed to input space mapping, output mapping is most effective when only the output of the coarse model needs to be realigned [499]. Similarly, an optimal choice for $f_{OM_g}(\cdot)$ and $f_{OM_d}(\cdot)$ depends on the quality of the coarse model. Again an identity mapping is used if no output mapping is selected by the algorithm, i.e., $f_{OM_g}(\cdot) = i_{g_{map}}$ and $f_{OM_d}(\cdot) = i_{d_{map}}$.

Difference Mapping The mapped output currents from (A.3) and (A.4) can further be corrected by applying difference mapping using controlled current and charge elements, in order to obtain the final output currents

$$i_g = i'_g + f_{DM_g}(v_g, v_d) \tag{A.5}$$

$$i_d = i'_d + f_{DM_d}(v_g, v_d) \tag{A.6}$$

where

$$f_{DM_g} = \Delta i_g(v_g, v_d) + \Delta \dot{Q}_g(v_g, v_d) \tag{A.7}$$

$$f_{DM_d} = \Delta i_d(v_g, v_d) + \Delta \dot{Q}_d(v_g, v_d) \tag{A.8}$$

In (A.7) and (A.8), Δi_g (Δi_d) and ΔQ_g (ΔQ_d) represent the current and charge corrections for the gate (drain) terminal, and $\Delta \dot{Q}_g$ and $\Delta \dot{Q}_d$ are the time derivatives of the correction charges. $f_{DM_g}(\cdot)$ and $f_{DM_d}(\cdot)$ represent the difference mapping equations and both are zero if no difference mapping exists. Difference mapping is most effective if the difference between the coarse model and the true data follows a predictable pattern [499].

There are many possible implementations for $f_{IM_g}(\cdot)$, $f_{IM_d}(\cdot)$, $f_{OM_g}(\cdot)$, $f_{OM_d}(\cdot)$, $f_{DM_g}(\cdot)$ and $f_{DM_d}(\cdot)$, ranging from simple linear mappings to powerful ANN-based mappings [230, 485]. As discussed in subsection A.2.1, the optimal choice will depend on the problem and available coarse model structures.

A.2.3.2 Equivalent Circuit Encoding and Solution Representation

Recall from Fig. A.1 that we assume a base circuit model with 3 nodes denoted by G, D, S and three connecting branches $B = \{GS, GD, DS\}$. Let $E_{b,b\in B}$ be the discrete variable representing the type of element present in a branch (e.g., $E = 1$: Diode, $E = 2$: Capacitor, etc). In addition, since we wish to explore different circuit topologies we allow multiple parallel branches for each of the 3 main intrinsic branches (not unlike the parallel augmentation used in [509]). Let p denote the number of such parallel branches. This means that a complete circuit with a maximum of p parallel branches for each main branch can be written as a set of three-tuples:

$$\{(E_{GS_1}, E_{GD_1}, E_{DS_1}), \ldots, (E_{GS_p}, E_{GD_p}, E_{DS_p})\} \tag{A.9}$$

This is illustrated graphically in Fig. A.3. As an example, the equivalent circuit model

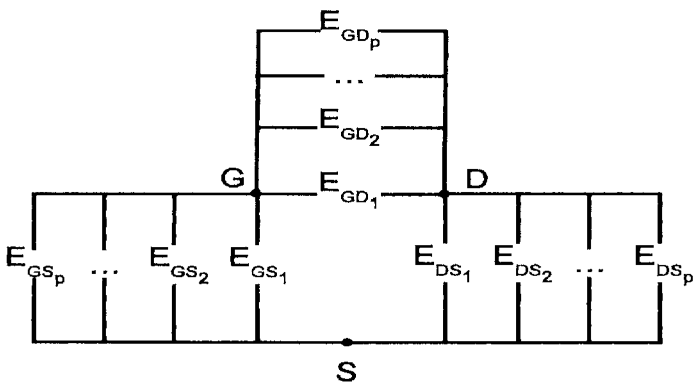


Figure A.3: Generalized base circuit model with p parallel branches connecting the base circuit nodes G, D and S.

shown in Fig. A.1 can be expressed by taking $p = 2$, two nonlinear capacitors for E_{GS_1} and E_{GD_2}, two diodes for E_{GS_2} and E_{GD_1}, a nonlinear source for E_{DS_1}, and using the respective formula for each element.

Besides different circuit topologies, we go beyond pure models where each element is approximated by the same type of equivalent circuit model. We wish to allow a hybrid equivalent circuit model where each element E_{b_r} $(b \in B, r = 1, .., p)$ may be approximated by a different equivalent circuit model type. Let M be a discrete variable that represents the model type (e.g., $M = 1$: Curtice [491], $M = 2$: Materka [493], etc.). The circuit representation then becomes:

$$\begin{aligned}&\{((M,C)_{GS_1},(M,C)_{GD_1},M_{DS_1}),\ldots,\\&((M,C)_{GS_p},(M,C)_{GD_p},M_{DS_p})\}\end{aligned} \tag{A.10}$$

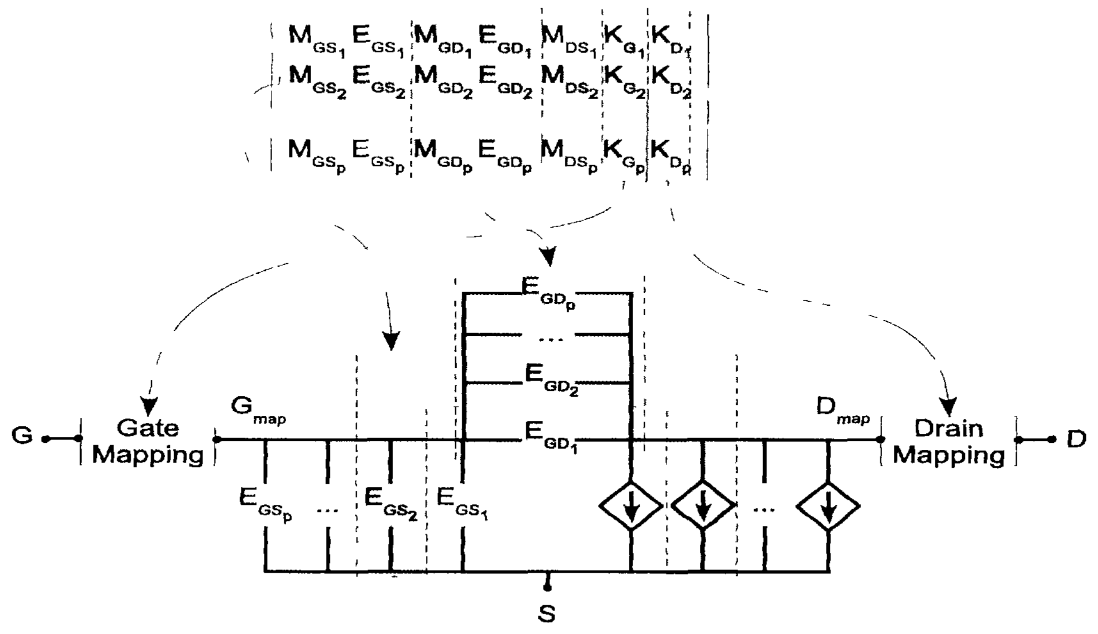
with the tuple $(M,C)_{b_r}$ representing the element E in parallel branch r of main branch $b \in B$ whose approximation is dictated by model type M. Since we assume the use of voltage controlled current sources for the DS branch therefore we can omit the element type parameter in the DS branch from the chromosome representation.

Finally, we come to the optimal choice of mapping structure which is applied to the circuit model as discussed so far. Let K denote the type of mapping used (e.g., $K = 1$: input mapping, $K = 2$: output mapping, etc.) and let the subscripts G and D indicate if the mapping is applied on the gate terminal or the drain terminal. If a mapping is not applied, this is denoted by $K = 0$. Together, this results in the following final circuit representation in a matrix form:

$$\left[\begin{array}{cc|cc|c|cc} M_{GS_1} & E_{GS_1} & M_{GD_1} & E_{GD_1} & M_{DS_1} & K_{G_1} & K_{D_1}\\ M_{GS_2} & E_{GS_2} & M_{GD_2} & E_{GD_2} & M_{DS_2} & K_{G_2} & K_{D_2}\\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\ M_{GS_p} & E_{GS_p} & M_{GD_p} & E_{GD_p} & M_{DS_p} & K_{G_p} & K_{D_p} \end{array}\right] \tag{A.11}$$

Note that this setup allows different mappings to be applied to the gate and drain separately and allows multiple mapping types to occur concurrently. The only restriction is that input mapping, output mapping, and difference mapping may only occur once in each mapping column (for example, applying output mapping twice on the drain terminal is not allowed). Absence of a certain branch or mapping is indicated by zeros at the corresponding locations. A graphical representation of the matrix in (A.11) is shown in Fig. A.4.

The matrix in equation (A.11) represents the chromosome of an individual, with M, C, K representing the different genes. In a real organism each gene has a number of alleles that can occupy each of the gene loci. The same is true here. In the concrete implementation, each of the symbols M, C, K is represented by a number that indicates the model type, the element type, and the mapping type, respectively. Table A.1 shows the different alleles used for each gene. Thus, we have reduced the problem to combinatorial optimization, the total number of combinations being:



M_{GS_1} E_{GS_1} M_{GD_1} E_{GD_1} M_{DS_1} K_{G_1} K_{D_1}
M_{GS_2} E_{GS_2} M_{GD_2} E_{GD_2} M_{DS_2} K_{G_2} K_{D_2}
M_{GS_p} E_{GS_p} M_{GD_p} E_{GD_p} M_{DS_p} K_{G_p} K_{D_p}
G
Gate Mapping
G_{map}
E_{GD_p}
...
E_{GD_2}
E_{GD_1}
D_{map}
Drain Mapping
D
E_{GS_p} ... E_{GS_2} E_{GS_1}
...
S

Figure A.4: Graphical representation of encoding an equivalent circuit model and mapping structures as a matrix. A mapping type can only occur once in each mapping column. If a branch or mapping is not present this is denoted by zeros at the corresponding locations.

$$\left[\sum_{l=1}^{p}\binom{(N_M\cdot N_E)+l-1}{l}\right]^2\cdot\left[\sum_{l=1}^{p}\binom{N_M+l-1}{l}\right]\cdot\left[\sum_{l=0}^{N_K}\binom{N_K}{l}\right]^2 \tag{A.12}$$

with N_M, N_E and N_K representing the number of different model types, element types, and mapping types, respectively. Thus each circuit generated by the genetic algorithm is encoded as a p-by-7 matrix.

	Chromosomal encoding
Model type (M)	1: Curtice, 2: Materka, 3: Chalmers
Element type (E)	1: Capacitor, 2: Diode
Mapping type (K)	1: Input mapping, 2: Output mapping 3: Difference mapping

Table A.1: Encoding table containing codes for each allele used in the chromosomal encoding.

An example of a knowledge circuit (without mapping structures) and its equivalent representation in the matrix form is shown in Fig. A.5. The circuit has a Chalmers capacitor and a Curtice diode in the *GS* branch and two Materka capacitors with different nonlinear coefficients, a Curtice diode, and a Chalmers diode in the *GD* branch. A Materka source in the *DS* branch completes the circuit. Such topological mixing of different equivalent circuit models allows for more freedom and flexibility to fit the device behavior with improved accuracy over traditional homogeneous equivalent circuit models and/or to permit a simpler mapping structure.

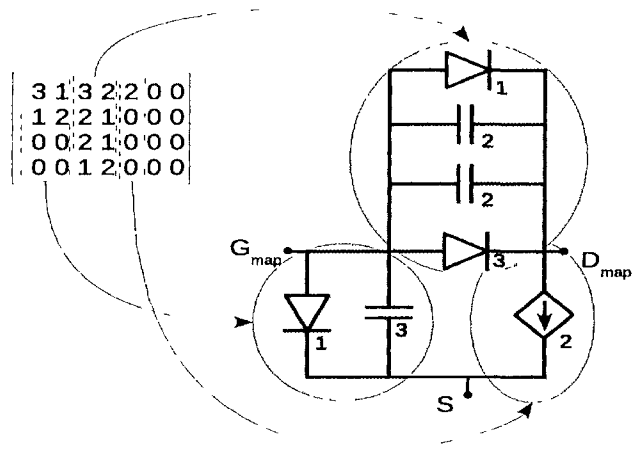


Figure A.5: Example of encoding a circuit topology into a chromosome using the proposed encoding from (A.11). *The numbers on the circuit represent the model type used (see Table A.1). The circuit has a Chalmers capacitor and a Curtice diode in the GS branch and two Materka capacitors with different nonlinear coefficients, a Curtice diode, and a Chalmers diode in the GD branch. A Materka source in the DS branch completes the circuit. Such hybrid of different model types allows flexibility of mapping and expands the search space for improved accuracy.*

An example with explicit mapping structures is depicted in Fig. A.6. In this example input mapping and output mapping are applied to the gate terminal and difference mapping is applied to the drain terminal.

A.2.4 Genetic Operators

Once an encoding scheme has been defined, it becomes possible to define the genetic operators. Recall from section A.2.2 that the genetic algorithm starts from a population of transistor models. Based on this population (namely the parents) the genetic algorithm generates a new population (namely the children) of transistor models based on two genetic operators: mutation and crossover. Which models are selected as parents is determined by a selection function. The selection function will take into account the fitness (defined as the model accuracy in terms of training error, described in section A.2.5) of each transistor model so that models with a better accuracy (or lower training error) have a higher probability of being selected as parents. Thus by iteratively applying the genetic operators on a starting population, selection should drive the population to the optimal solution. So an evolutionary algorithm is completely determined by its encoding and genetic operators. The next subsections formulate the genetic operators defined in this work.

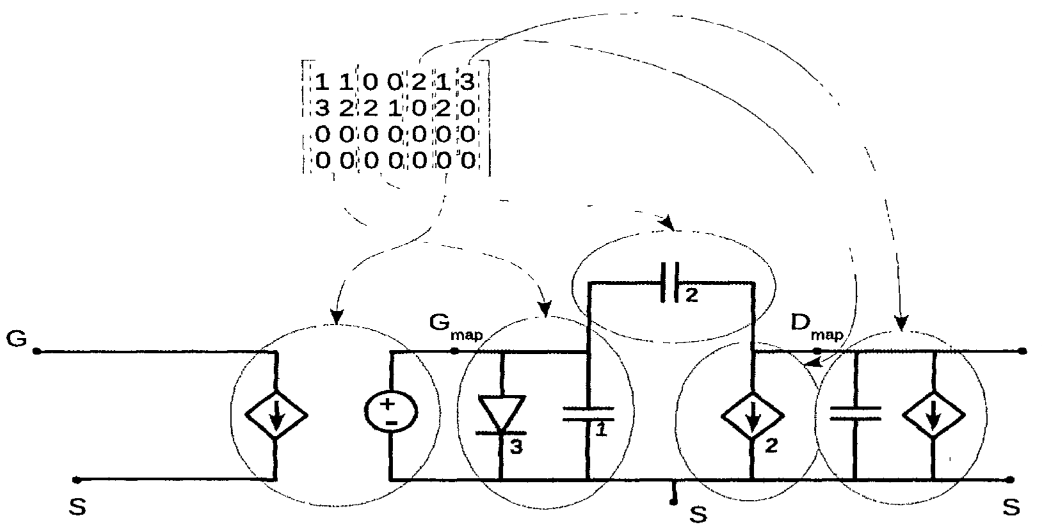


Figure A.6: Circuit encoding example with mapping structures. Input and output mappings are applied to the gate terminal, while difference mapping is applied to the drain terminal.

A.2.4.1 Mutation

The purpose of the mutation operator is to ensure that every region of the search space can be reached. It should therefore contain sufficient randomness to make this possible. We employ a meta-mutation operator that selects one or more simple mutation operators according to various probabilities. Let X be the matrix in (A.11) which represents a particular encoded circuit. The simple mutation operators are formulated as follows:

- *Delete a random row*: a row r $(1 \leq r \leq p)$ is chosen randomly from X and replaced by a row of zeroes. This has the effect of deleting one parallel branch from each of the three main circuit branches (GD, GS, DS). This operator is illustrated in Fig. A.7.

- *Replace a random row*: a row r is chosen randomly from X and replaced by a randomly generated vector $[M_{GS_r}, E_{GS_r} M_{GD_r}, E_{GD_r}, M_{DS_r}, K_{G_r}, K_{D_r}]$. This will add or replace an existing mapping structure or parallel branch.

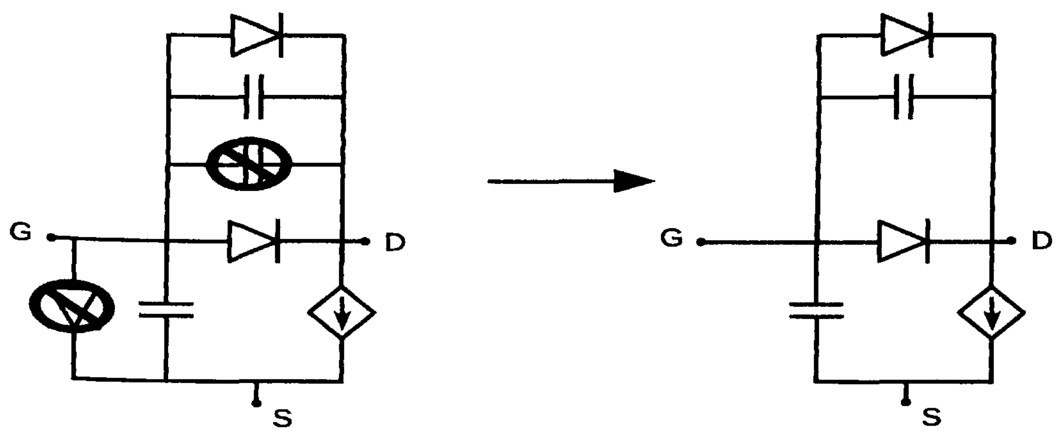
- *Rotate the model types*: the non-zero model types $[M_{GS_r}, M_{GD_r}, M_{DS_r}]$ of each row r of X are permuted one clockwise cycle resulting in $[M_{DS_r}, M_{GS_r}, M_{GD_r}]$. For example, if this operator is applied to the circuit in Fig. A.5 the GS branch will contain a Materka capacitor (type inherited from the original Materka source in the DS branch) and Materka diode (from the Materka capacitor in the second parallel branch of GD) instead of a Chalmers capacitor and Curtice diode.

- *Rotate the element types*: the same as the previous operator only now applied to the element types in the GS and GD branches.

- *Swap the mapping structures*: the mapping structures which are applied to the drain terminal are switched to the gate terminal, and vice versa. Thus K_{D_r} swaps places with K_{G_r} for each row r of X. An illustration is given in Fig. A.8. First input mapping and output mapping were applied on the gate terminal and difference mapping on the drain terminal. After applying the mutation operator the situation is reversed.

- *Delete a random branch*: a row r and branch b from X is chosen randomly, uniquely identifying a particular parallel branch, i.e., a tuple (M_{b_r}, E_{b_r}). This branch is then deleted from the circuit. This procedure is repeated three times.

- *Replace a random branch*: the same as the previous operator instead now the randomly selected branch is replaced by a randomly generated one. If the selected branch was empty (consisted of zeroes) this amounts to adding a new parallel branch. This is repeated twice.



$$\begin{bmatrix} 3\,1\,3\,2\,2\,0\,0 \\ \cancel{1\,2\,2\,1\,0\,0\,0} \\ 0\,0\,2\,1\,0\,0\,0 \\ 0\,0\,1\,2\,0\,0\,0 \end{bmatrix} \qquad \begin{bmatrix} 3\,1\,3\,2\,2\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,2\,1\,0\,0\,0 \\ 0\,0\,1\,2\,0\,0\,0 \end{bmatrix}$$

Figure A.7: Example of applying the "Delete a random row" *mutation operator, resulting in eliminating the diode in the GS branch and the capacitor in the GD branch.*

Which subset of operators is applied on a given individual will depend on the random numbers generated uniformly. The advantage of this approach is that it still allows for large jumps in the search space without completely destroying the individual in one step (as would be the case if one single, complex mutation operator was used). Note that the collection of simple operators edge on the conservative side when it comes to circuit complexity. Three of the operators leave the size of the circuit unchanged, two of the operators remove branches, while two potentially add new elements. This helps ensure that the genetic algorithm does not needlessly generate overly complex circuits.

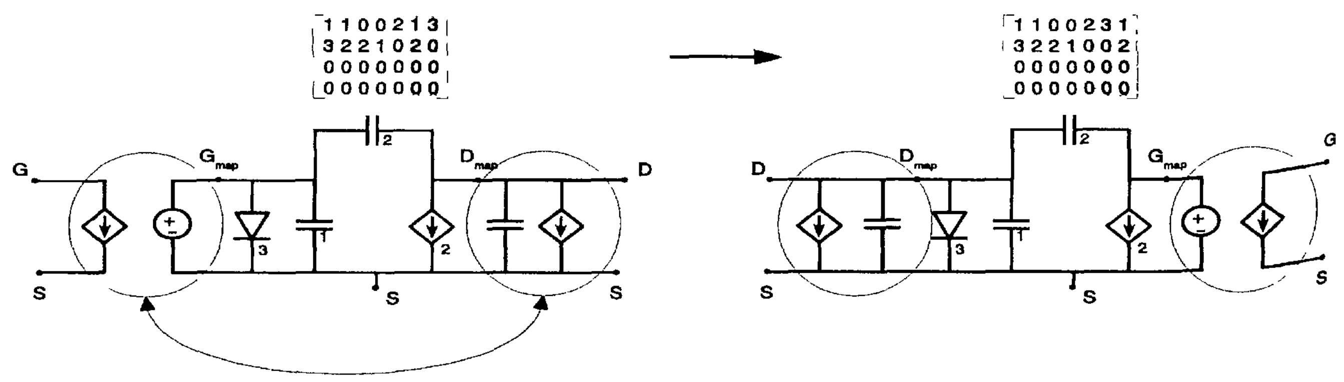


Figure A.8: Example of applying the "Swap the mapping structures" *mutation operator. The mappings that applied to the gate terminal are transferred to the drain terminal and vice versa. In the figure this means that first input mapping and output mapping were applied on the gate and difference mapping on the drain. After applying the mutation operator the situation is reversed.*

A.2.4.2 Crossover

The purpose of the crossover operator is to recombine genetic information from two parents in order to produce one or more offspring. It is important to find the right balance between exploration and exploitation in the recombination operator. The offspring should contain recognizable genetic information from both parents. If the offspring is too similar to either one of the parents, the genetic algorithm may concentrate too much on exploitation and less on exploration. If the offspring contains too much randomness, the converse is true.

We again use a combination of different simple recombination operators that act on two parents X_1 and X_2, both encoded using (A.11):

- *Swap the models*: the model type sub-matrix of X_1,

$$M_{X_1} = \begin{bmatrix} M_{GS_1} & M_{GD_1} & M_{DS_1} \\ M_{GS_2} & M_{GD_2} & M_{DS_2} \\ \dots & \dots & \dots \\ M_{GS_p} & M_{GD_p} & M_{DS_p} \end{bmatrix} \tag{A.13}$$

  replaces the equivalent sub-matrix M_{X_2} of X_2 and vice versa. To prevent orphaned model types only non-zero elements are replaced. As an example, this means a parent circuit which contains a mixture of Materka and Curtice elements in its *GD* branch will result in a child circuit with the exact same elements but now approximated by the Chalmers equations (if we assume the other parent only has Chalmers elements in its *GD* branch).

- *Swap the elements*: this operation is equivalent to the previous operator except this time only element type information is exchanged. The relevant sub-matrix

$E_{X_{1,2}}$ is then

$$E_{X_{1,2}} = \begin{bmatrix} E_{GS_1} & E_{GD_1} \\ E_{GS_2} & E_{GD_2} \\ \dots & \dots \\ E_{GD_p} & E_{GD_p} \end{bmatrix} \tag{A.14}$$

Thus now the element types are changed while the model types remain fixed. For example, application to a circuit with two Materka diodes in a certain branch may result in a circuit with two Materka capacitors in that branch.

- *Swap the mapping structures*: analogous to the previous operators the mapping structure sub-matrix $K_{X_{1,2}}$ is exchanged

$$K_{X_{1,2}} = \begin{bmatrix} K_{G_1} & K_{D_1} \\ K_{G_2} & K_{D_2} \\ \dots & \dots \\ K_{G_p} & K_{D_p} \end{bmatrix} \tag{A.15}$$

  As an example, by applying this operator, the mapping structures that surrounded the equivalent circuit model of X_1 are transferred to the equivalent circuit of X_2. Likewise, the mapping structures that surrounded the equivalent circuit of X_2 are transferred to the equivalent circuit of X_1. The mapping information is effectively exchanged leading to two possible offspring. If neither parent contains a mapping then this operator falls back to swapping the models.

- *One-point row crossover*: classic one-point-crossover is performed on the matrix rows. A row r is selected randomly and an offspring is generated by taking rows 1 to $r-1$ from X_1 together with rows r to p from X_2. A second offspring can be generated by reversing the roles of X_1 and X_2. In circuit terms this means that the outer $p-r$ parallel branches of the parent circuits are exchanged in order to generate two offspring.

- *Exchange a branch*: a branch $b \in \{GS, GD, DS\}$ is chosen randomly and exchanged between both parent circuits with all attached parallel branches.

However, in contrast to the mutation operator, only a single recombination operator can be active at a time. Fig. A.9 illustrates the application of the branch exchange operator (only one of the two possible children is shown).

A.2.4.3 Repair function

Finally, the reader may have noted that without further modifications, the algorithm as described so far may generate invalid circuits. Therefore at the end of each genetic operator an extra check on the circuit topology is performed, and the individual is repaired if necessary. For example, if the circuit is not closed, a random branch is added to close the circuit.

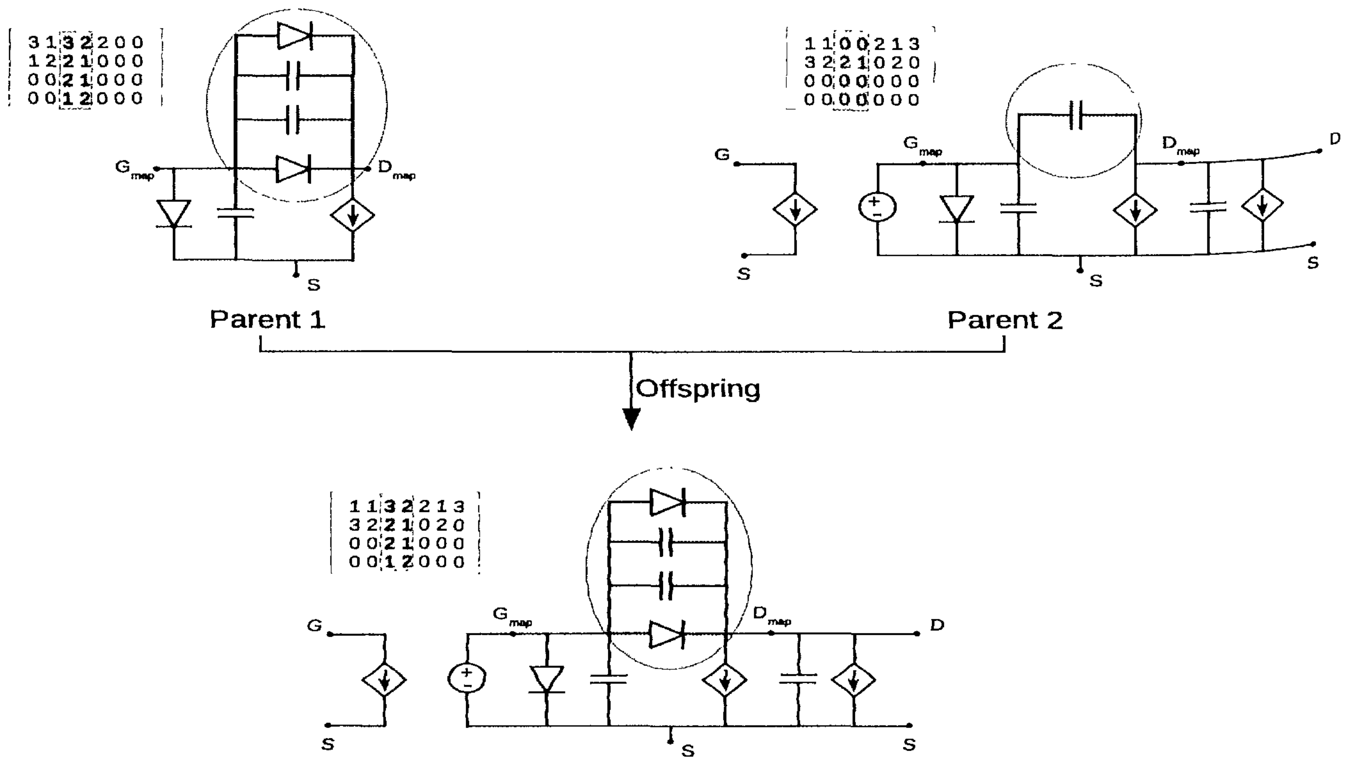


Figure A.9: Example of applying the "Exchange a branch" *crossover operator. In this case the GD branch is randomly chosen and exchanged between the two parents (including all parallel branches). Only one of the two possible children is shown in the figure: the child which inherits everything form the parent on the right, except the GD branch, which it inherits from the parent on the left. This is equivalent to exchanging the* 2^{nd} *and* 3^{rd} *columns in the matrix encoding.*

A.2.5 Fitness Function

A.2.5.1 Decoding a solution matrix into a circuit netlist

The fitness function is the core of the genetic algorithm. It maps an individual from the population onto a scalar score which represents the difference between the model and the transistor device data. A lower fitness implies a more accurate circuit model. In doing so, it must decode the matrix representing the individual into an equivalent circuit model. To make this possible, the fitness function uses an eXtensible Markup Language (XML) [510] file that contains the approximation equations (in a netlist compatible format) for each possible element that can occur in the circuit.

Fig. A.10 shows an example of two elements the file may contain. The first element contains the equation of a capacitor (which occurs in the *GS* branch) approximated by the Chalmers model. The second element shown in Fig. A.10 holds the equations for a linear difference mapping on the gate terminal. Thus, as is clear from the figure, XML is a markup language (i.e., it annotates existing data) that permits standardized representation of structured data. The choice for XML was natural since it is the defacto standard for structured data representation, and support for it is built-in to virtually every programming language and development environment. It also allows for easy extension of this approach to new equivalent circuit models and mapping types in the future.

```
</NetlistElements>

  <Element model="Chalmers" branch="GS" type="Capacitor">
    <SDD>SDD:SDD2P3  Gc1 Sc1 Dc1 Sc1  I[1,1]=Qgs(_v1, _v2)  I[2,0]=0</SDD>
    <Parameters>
      P11 = 0.423083438 opt{ unconst };
      P21 = 0.012810991 opt{ unconst };
      Cgs0 = 6.75906440e-13 opt{ unconst };
    </Parameters>
    <Functions>
      Qgs(v1, v2) = Cgs0*(P11*v1+ln(cosh(P11*v1)))*(1+tanh(P21*v2))/P11
    </Functions>
  </Element>

  ...

  <Mapping terminal="Gate" type="DiffMapping">
    <Parameters>
      ag0 = 0 opt{  +/- 5 };
      ag1 = 0 opt{  +/- 5 };
      ag2 = 0 opt{  +/- 5 };
      bg0 = 0 opt{  +/- 5 };
      bg1 = 0 opt{  +/- 5 };
      bg2 = 0 opt{  +/- 5 };
    </Parameters>

    <Functions>
      Ig(Vg, Vd) = ag0+ag1*Vg+ag2*Vd;
      Qg(Vg, Vd) = bg0+bg1*Vg+bg2*Vd;
    </Functions>
  </Mapping>
  ...
</NetlistElements>
```

Figure A.10: Example XML fragment for storing the implementation of each of the elements that may appear in a circuit. The approximation for a circuit element/mapping is split into different parts: the actual equation (the <Functions> *tag), where the equation is used in the circuit netlist (*<SDD> *tag), and the parameters that occur in the equation (*<Parameters> *tag). The* opt{...} *specifiers indicate to the circuit simulator that will process this netlist that those parameters are optimizable within the given bounds.*

Furthermore, once data information is stored in XML format it becomes very easy to manipulate and extend. In particular querying the XML data for specific information can be easily done. This is made possible through the XML Path Language (XPath) [511]. For example, if one wanted to uniquely retrieve the first element listed in Fig. A.10, one would perform the following XPath query:

//**Element**[@**model**=*'Chalmers'*] and [@**branch**=*'GS'*] and [@**type**=*'Capacitor'*]

Thus a matrix that represents an encoded circuit based on (A.11) can be easily mapped onto a series of XPath calls that collect the necessary elements that make-up the circuit. These elements are then combined and merged into a base template netlist in order to arrive at the final netlist that fully implements the circuit. This netlist can then be executed by the circuit simulator.

A.2.5.2 Mapping a circuit to a fitness value

The next step is to produce a scalar fitness value for a given netlist. Let N be a netlist that implements a particular circuit. N is the result of decoding an encoded matrix X according to the procedure described above. The accuracy of N as a model of the transistor in question, is calculated based on the training data, such as the DC and bias-dependent S-parameter data. Let this data be denoted by $\mathbf{y}$ and the number of data samples by n. In addition, the empirical equations in each branch of the equivalent circuit model contain a number of parameters that must be set (e.g., the P_{11} and P_{21} parameters from Fig. A.10). Let θ_E denote the values of these parameters and let N_{θ_E} be the circuit represented by netlist N whose circuit parameters have been set to θ_E. Let $\mathbf{g}(\cdot)$ be a function that represents the execution of the netlist by the circuit simulator. $\tilde{\mathbf{y}} = \mathbf{g}(N_{\theta_E})$ then denotes the prediction of the training data by N_{θ_E}. In order to achieve maximum accuracy, θ_E must be optimized to an optimal value θ_E^*. This can be performed by a circuit simulator (in this paper we use Agilent Advanced Design System (ADS) [512]). The optimization routines present in the circuit simulator will optimize θ_E in order to accurately fit the empirical equivalent circuit model to the training data. Denote this optimized circuit model by $N_{\theta_E^*}$ with

$$\theta_E^* = \arg\min_{\theta_E} h(\mathbf{y}, \mathbf{g}(N_{\theta_E})) \tag{A.16}$$

The score returned by the function $h(\cdot)$, and minimized by the circuit simulator, is a weighted sum (using weights w_i with $||\mathbf{w}|| = 1$) of the absolute error on the DC and the real and imaginary parts of the S-parameters, i.e.

$$h(\mathbf{y}, \tilde{\mathbf{y}}) = \sum_{i=1}^{9} w_i \cdot ASE(y_{o_i}, \tilde{y_{o_i}}) \tag{A.17}$$

Where y_{o_i} and $\tilde{y_{o_i}}$ represent the true and predicted values of each output, i.e., the drain current I_d, and the real and imaginary S-parameters ($o_i, \tilde{o}_i \in \{I_d, real(S_{11}), imag(S_{11}), real(S_{12}), imag(S_{12}), real(S_{21}), imag(S_{21}), real(S_{22}), imag(S_{22})\}$). The Average Scaled Error (ASE) between y_{o_i} and $\tilde{y_{o_i}}$ is defined as

$$ASE(y_{o_i}, \tilde{y_{o_i}}) = \frac{1}{n} \sum_{k=1}^{n} \frac{|y_{o_{i_k}} - \tilde{y_{o_{i_k}}}|}{|\max(y_{o_i}) - \min(y_{o_i})|} \tag{A.18}$$

where i, $i = 1, 2, ..., 9$ denotes the index of outputs, and k, $k = 1, 2, ..., n$ denotes the index of the data samples.

Note that so far we have only discussed the optimization of the parameters in the equivalent circuit models themselves. If a mapping structure is present in an individual solution the parameters of that mapping (denoted by θ_M) need to be optimized as well (e.g., the $ag_0, ..., bg_0$ parameters from Fig. A.10). However, the effective use of mapping structures requires that the circuit without mapping is already of reasonable quality [483]. Therefore, if a solution contains one or more mapping structures, two optimizations are performed. First the circuit is optimized over θ_E without the mapping structure in order to find $N_{\theta_E^*}$ as in (A.16). Then, for the fixed value of θ_E^* an optimization is performed over θ_M to find the optimal set of mapping parameters θ_M^*:

$$\theta_M^* = \arg\min_{\theta_M} h(\mathbf{y}, \mathbf{g}(N_{\{\theta_E^*, \theta_M\}})) \tag{A.19}$$

Let $N_{\{\theta_E^*, \theta_M^*\}}$ be the netlist N whose parameters are set to these optimal values. The final fitness value returned for a given netlist N is thus

$$fitness(N) = h(\mathbf{y}, \mathbf{g}(N_{\{\theta_E^*, \theta_M^*\}})) \tag{A.20}$$

The full search by the genetic algorithm in order to find the optimal circuit topology with optimal mapping structure N^* can then be written as

$$N^* = \arg\min_{N} fitness(N) \tag{A.21}$$

This whole process is illustrated graphically in in Fig. A.11.

A.2.5.3 Parallel execution of circuit simulations

The repeated optimization of each solution generated by the genetic algorithm can be expected to be computationally expensive. Luckily, a major advantage of evolutionary algorithms is that they naturally allow for parallelization. The parallel computing model is Single-Process-Multiple-Data thus the fitness of each circuit in the population can be calculated independently on different CPU's. Thus the fitness function is implemented so that multiple circuit simulator instances can be run in parallel, each simulating a particular circuit of the population.

In addition a fitness cache is used to prevent running the same simulations twice (through selection and elitism the population may contain duplicates), reducing the running time even further. This can be done since the simulations are deterministic.

A.3 Examples

A.3.1 Evolutionary Modeling of a GaAs MESFET

This example illustrates the proposed technique on a large-signal FET model trained with both DC and bias-dependent S-parameter data. The fine device data is gener-

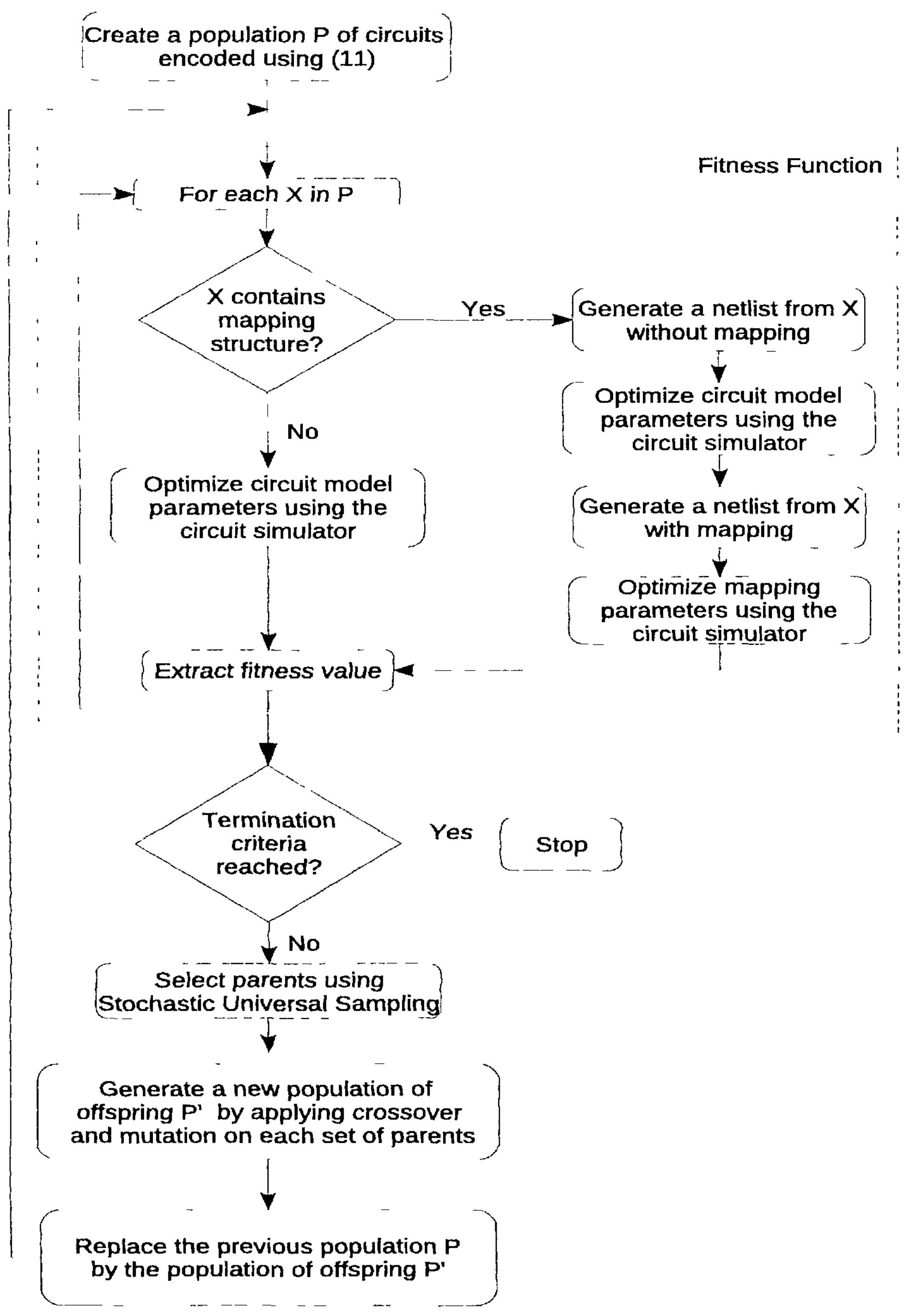

Create a population P of circuits encoded using (11)
For each X in P
Fitness Function
X contains mapping structure?
Yes
Generate a netlist from X without mapping
Optimize circuit model parameters using the circuit simulator
Generate a netlist from X with mapping
Optimize mapping parameters using the circuit simulator
No
Optimize circuit model parameters using the circuit simulator
Extract fitness value
Termination criteria reached?
Yes
Stop
No
Select parents using Stochastic Universal Sampling
Generate a new population of offspring P' by applying crossover and mutation on each set of parents
Replace the previous population P by the population of offspring P'

Figure A.11: Flowchart for the evolution of optimal knowledge-based models.

ated using an ADS internal GaAs FET model [230, 492]. The data is available at 20 frequencies (1-20 GHz) and 125 biases ($V_g \in [-1,0]$ V, $V_d \in [0.2,5]$ V).

The equivalent circuit models used as building blocks for the genetic algorithm are the Curtice model [491], Materka model [493], and the Chalmers model [494]. The mapping structures included are input mapping, output mapping, and difference mapping. For this paper we build upon the neuro-space mapping approach described in

[230] and [485]. However, now the focus is on type selection of the mapping structure while at the same time achieving best possible quality of the equivalent circuit. This is made possible by mixing model types and performing circuit topology optimization. We wish to keep the mapping structure simple to make the model more efficient and robust. For this reason we use a linear ANN for f_{IM} and f_{OM} which is a simpler version of the ANN-based mapping used in [230]. Of course a nonlinear ANN-based mapping can still be included if necessary, there is no inherent limitation in the methodology, it will only improve the results.

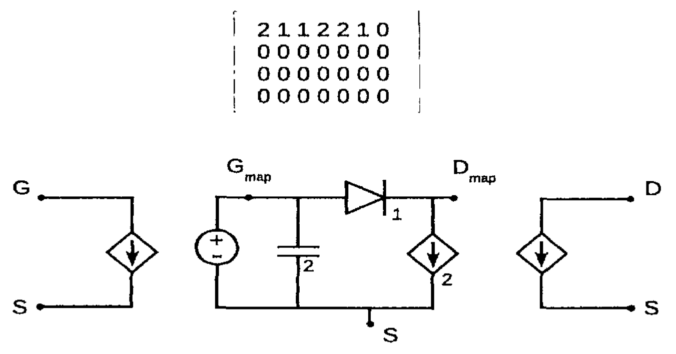
The maximum number of parallel branches (p) was set to 4. If we then calculate the number of possible circuits (i.e., the size of the search space) according to (A.12) we see that this is more than 9.50×10^7. Far too large to explore manually.

All tests were run on a Matlab 7.8 R2009a platform [513] utilizing ADS 2006 [512] and the Matlab Genetic Algorithm & Direct Search, and parallel computing toolboxes. Simulations were run on a Quad Core Intel machine with 2GB main memory running Ubuntu Linux 8.10. Using the local scheduler from the parallel computing toolbox, this allowed for running 4 ADS simulations in parallel. If also the distributed scheduler is available, the implementation can also seamlessly run on a large cluster or grid of machines, without requiring any modifications to the code.

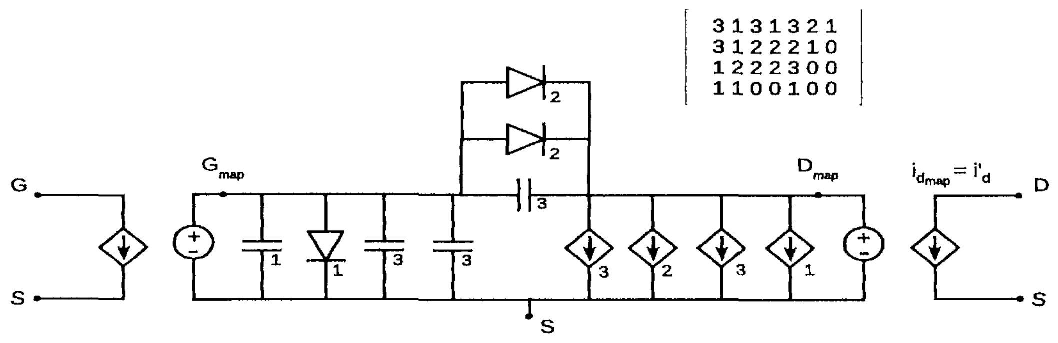
During the execution of the genetic algorithm, ADS was configured to perform 10 optimization iterations using the standard gradient-based method per circuit fitness evaluation. The time for one fitness evaluation is roughly 180 seconds on average for an individual with no knowledge and double that if one or more mapping structures are present. The initial values for each of model parameters were set to good, sensible values. The final solution was optimized for an additional 200 iterations. If ADS failed to converge during the model parameter optimization a score of 1000 was assigned to the individual causing the failure. If the failure occurred during the knowledge optimization the score of the individual without knowledge is returned. The population size was set to 20, the maximum number of generations to 100, the crossover probability to 0.8, and 3 elite individuals were used. The initial population consists of a pure equivalent circuit model of each type (Curtice, Chalmers, Materka) with $p = 2$ and randomly generated circuits, also with $p = 2$. The selection function utilized is Stochastic Universal Sampling [317]. Once the evolution has terminated, we record the Average Scaled Error and Maximum Absolute Error (MAE) for the best solution found. Both error metrics are calculated between the device data and the predicted model response over the DC and S-parameter responses. The MAE between the true values $\mathbf{y}$ and the predicted values $\tilde{\mathbf{y}}$ is defined as

$$MAE(\mathbf{y}, \tilde{\mathbf{y}}) = \max(|\mathbf{y} - \tilde{\mathbf{y}}|) \tag{A.22}$$

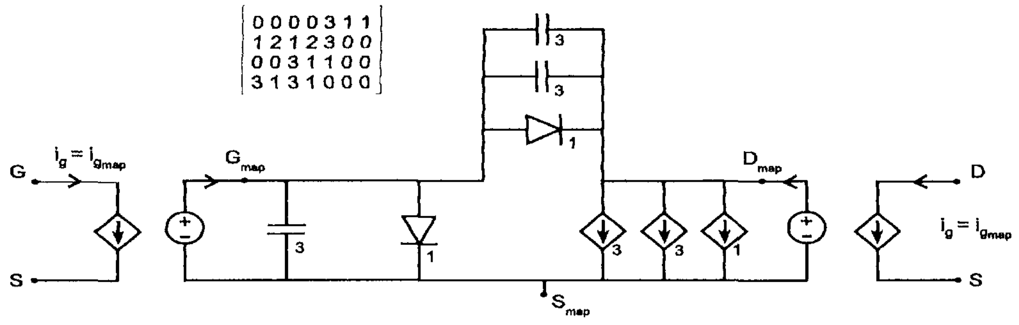
For the MESFET problem the final solution and two intermediate solutions generated by the genetic algorithm are shown in Fig. A.12. The best solution achieved by the genetic algorithm is a hybrid model that mixes the Curtice and Chalmers models



(a) Initial solution from the first generation (fitness = 0.171)



(b) Intermediate solution (fitness = 0.010)



(c) Final solution after 100 generations (fitness = 0.001)

Figure A.12: Genetic Algorithm generated solutions for the MESFET example.

with suitable mapping structures:

- Mapping: Input mapping on the gate terminal and drain terminal
- *GS* branch: Curtice Diode, Chalmers Capacitor
- *GD* branch: Curtice Diode, Chalmers Capacitor, Chalmers Capacitor
- *DS* branch: Chalmers Source, Chalmers Source, Curtice Source

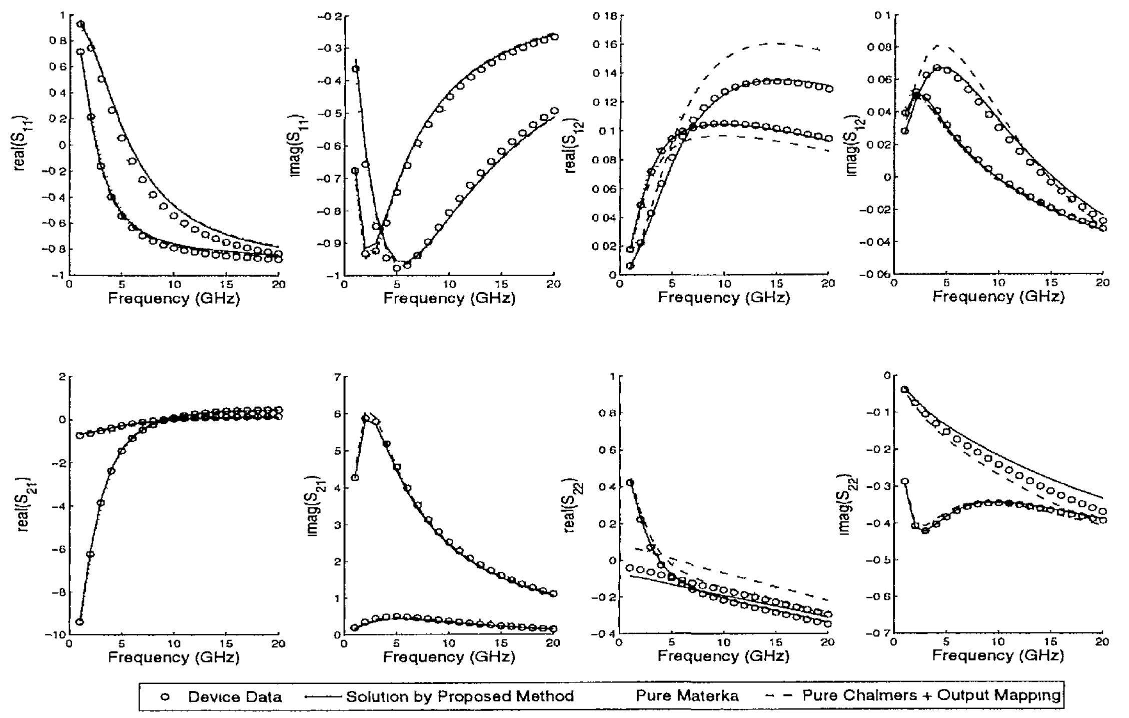


Figure A.13: S-parameter plot for $(V_g, V_d) \in \{(-1,0),(0.2,5)\}$ for the MESFET example.

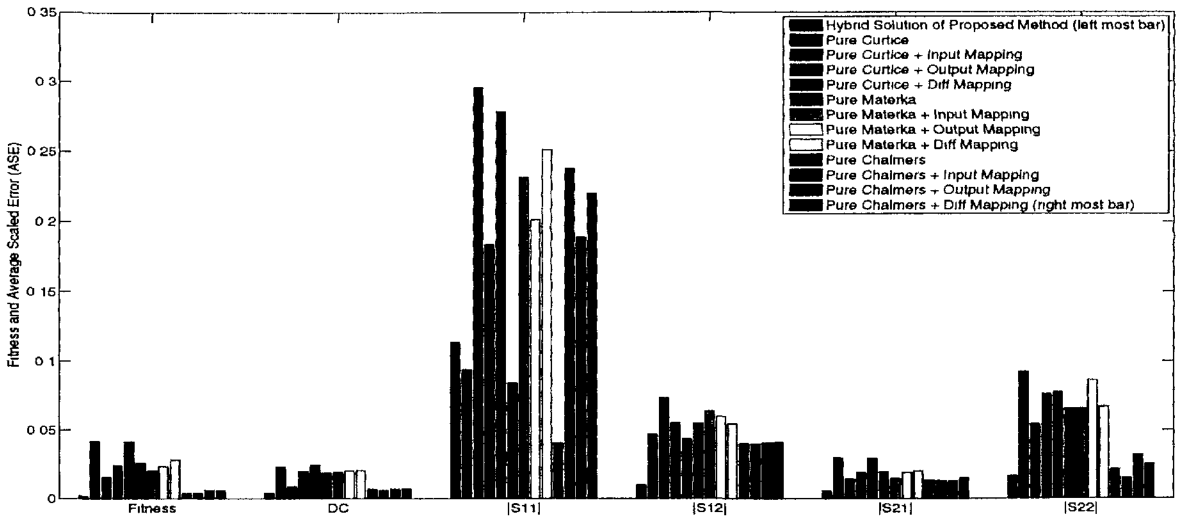


Figure A.14: Comparison of the proposed evolutionary modeling method with pure models on the MESFET modeling problem (Average Scaled Error). A lower fitness means that the overall error of DC and S-parameters is lower and that the model is more accurate. The solution from the proposed method is a hybrid model with mappings and gives the best accuracy overall.

Note that we have duplicate model elements in the *GD* and *DS* branches. Although these elements have the same type, their parameters are different (due to the optimiza-

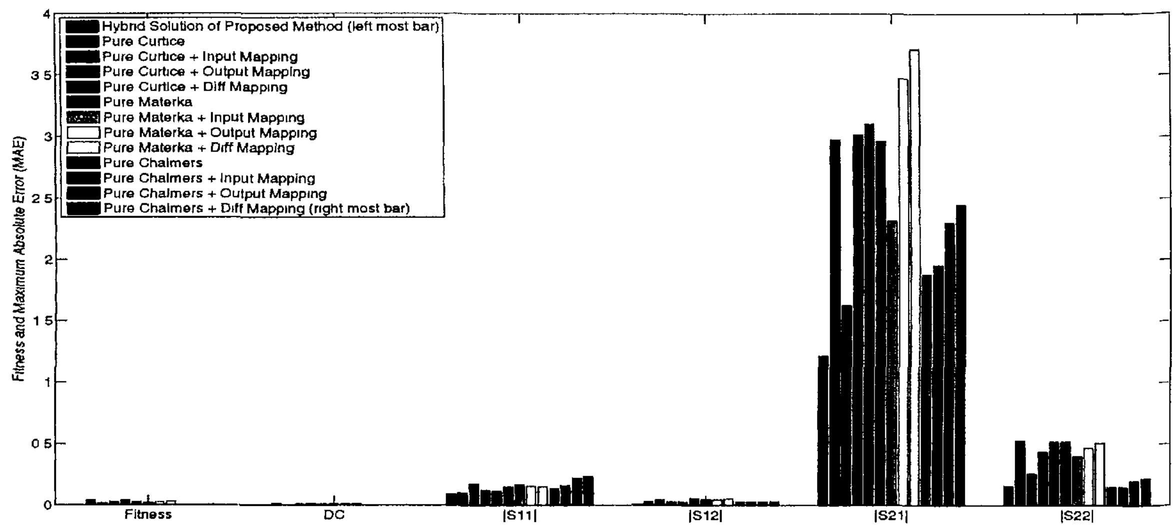


Figure A.15: Comparison of the proposed evolutionary modeling method with pure models on the MESFET modeling problem (Maximum Absolute Error). A lower fitness value means an overall more accurate model. The solution from the proposed method, which is a hybrid model with mappings, gives the best accuracy.

tion) thus their behavior is not the same.

For purposes of comparison the data was also modeled using pure forms of each of the different equivalent circuit model types. With pure models mean equivalent circuit models whose implementing equations all derive from the same model type and who have at most one mapping type applied (e.g., the standard Chalmers model with input mapping on both gate and drain). Fig. A.13 shows a plot of the S-parameters at 2 specific bias values for the genetic algorithm solution and two other models. From the figure it can be observed that a pure Materka model with input mapping performs better than a pure Curtice model. A pure Chalmers model with output mapping performs even better.

Fig. A.14 shows how the evolved solution compares to each of the pure models (the exact numbers can be found in Table A.2). As can be seen from Fig. A.14, the circuit generated by the genetic algorithm is the best solution (lowest fitness) over all other types of models under test. Regarding the average error it performs best on DC, S_{12}, S_{21} and gives the same performance as the best pure model on S_{22}. The lesser performance on S_{11} turns out to be due to a lower accuracy on the real part of S_{11} that gets magnified when calculating the Average Scaled Error of the magnitude (the generated solution is the best solution of the imaginary component). Fig. A.14 shows the accuracy in terms of average error. Worst case performance is shown in Fig. A.15 and the general tendency is the same. The solution found by the genetic algorithm performs equal to (DC,S_{22}) or better than (S_{11},S_{12},S_{21}) any of the pure models. Especially for S_{21} the worst case accuracy of the proposed solution is significantly better.

		Average Scaled Error (ASE)				
	Fitness	**DC**	**|S11|**	**|S12|**	**|S21|**	**|S22|**
Solution of Proposed Method	**0.0019**	**0.0040**	0.1133	**0.0099**	**0.0055**	0.0168
Pure Curtice	0.0422	0.0231	0.0941	0.0471	0.0300	0.0926
Pure Curtice + Input Mapping	0.0159	0.0087	0.2958	0.0733	0.0144	0.0547
Pure Curtice + Output Mapping	0.0244	0.0200	0.1838	0.0554	0.0191	0.0763
Pure Curtice + Diff. Mapping	0.0419	0.0245	0.2785	0.0438	0.0294	0.0777
Pure Materka	0.0261	0.0190	0.0845	0.0550	0.0196	0.0661
Pure Materka + Input Mapping	0.0205	0.0195	0.2315	0.0637	0.0149	0.0660
Pure Materka + Output Mapping	0.0235	0.0202	0.2012	0.0600	0.0189	0.0863
Pure Materka + Diff. Mapping	0.0285	0.0206	0.2512	0.0548	0.0200	0.0672
Pure Chalmers	0.0040	0.0070	**0.0410**	0.0401	0.0136	0.0212
Pure Chalmers + Input Mapping	0.0039	0.0062	0.2379	0.0400	0.0133	**0.0151**
Pure Chalmers + Output Mapping	0.0061	0.0071	0.1886	0.0408	0.0129	0.0321
Pure Chalmers + Diff. Mapping	0.0059	0.0073	0.2199	0.0413	0.0151	0.0255

		Maximum Absolute Error (MAE)				
	Fitness	**DC**	**|S11|**	**|S12|**	**|S21|**	**|S22|**
Solution of Proposed Method	**0.0019**	0.0043	**0.0891**	**0.0087**	**1.2154**	0.1583
Pure Curtice	0.0422	0.0132	0.1028	0.0323	2.9773	0.5257
Pure Curtice + Input Mapping	0.0159	0.0055	0.1733	0.0448	1.6263	0.2616
Pure Curtice + Output Mapping	0.0244	0.0090	0.1241	0.0294	3.0194	0.4360
Pure Curtice + Diff. Mapping	0.0419	0.0132	0.1147	0.0298	3.1051	0.5182
Pure Materka	0.0261	0.0116	0.1483	0.0557	2.9645	0.5163
Pure Materka + Input Mapping	0.0205	0.0100	0.1678	0.0519	2.3200	0.4023
Pure Materka + Output Mapping	0.0235	0.0105	0.1555	0.0468	3.4738	0.4683
Pure Materka + Diff. Mapping	0.0285	0.0139	0.1510	0.0533	3.7128	0.5077
Pure Chalmers	0.0040	**0.0027**	0.1362	0.0332	1.8802	0.1494
Pure Chalmers + Input Mapping	0.0039	0.0044	0.1631	0.0327	1.9553	**0.1480**
Pure Chalmers + Output Mapping	0.0061	0.0034	0.2208	0.0275	2.3055	0.1995
Pure Chalmers + Diff. Mapping	0.0059	0.0044	0.2362	0.0324	2.4523	0.2176

Table A.2: Comparison of the proposed evolutionary modeling method with pure models on the MESFET modeling problem. A lower fitness values means an overall more accurate model. The solution from the proposed method is a hybrid model with mappings and gives the best accuracy overall.

A.3.2 Evolutionary Modeling of a HEMT Device

The High Electron Mobility Transistor (HEMT) device is important in high frequency circuit design. In this example, the proposed technique is used to learn physics-based data of a HEMT device [230], with the training data (DC and bias dependent S-parameter data) generated from a physics-based device simulator, MINIMOS [514], by solving the device Poisson equations. The data is available at 40 frequencies (1-40GHz) and 125 biases ($V_g \in [-5,-1]$ V, $V_d \in [0,3]$ V).

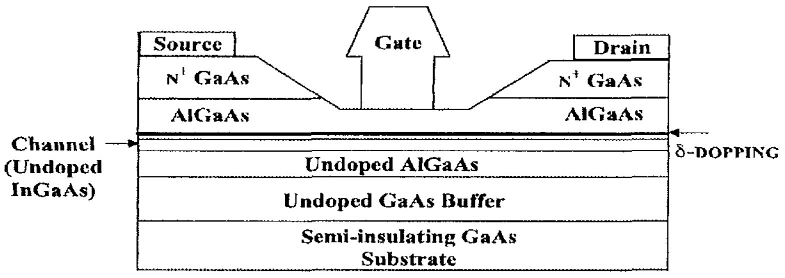


Figure A.16: Physical structure of a HEMT device used for generating fine data in the MINIMOS physics based device simulator.

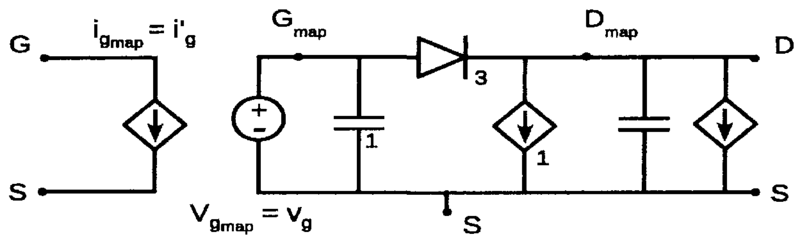
The HEMT structure used in setting up the physics-based simulator is shown in Fig. A.16. Since this is a more complex modeling problem we also add the necessary extrinsic components, though they remain fixed throughout the evolution. The same configuration (chomosomal encoding, genetic algorithm settings, ADS settings) as the first example is used.

The final solution and two intermediate solutions found by the genetic algorithm are depicted in Fig. A.17. The final solution is a hybrid model with mapping and is made up of the following elements:

- Mapping: Input mapping on the gate, input mapping and output mapping on the drain
- *GS* branch: Chalmers Capacitor, Curtice Capacitor
- *GD* branch: Materka Diode, Curtice Diode
- *DS* branch: Materka Source, Materka Source

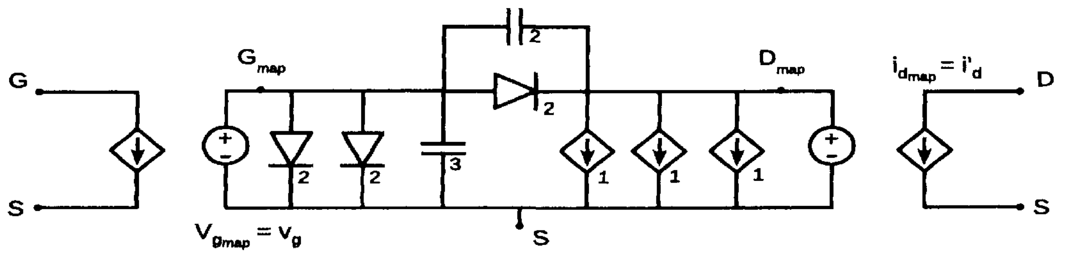
The S-parameter plot for this problem is shown in Fig. A.18. The genetic algorithm generated solution shows clear improvement over the other pure equivalent circuit models. Figures A.19 and A.20 show how the accuracy compares with each of the pure models and models with pre-determined knowledge. The exact numbers can be found in Table A.3. Again we see that the genetic algorithm solution compares favorably to the other pure models, both in worst case error and in average error. From Fig. A.19 it is interesting to note that eventhough pure Curtice and pure Materka models have high errors on S_{11}, the solution generated by the genetic algorithm performs much better while consisting mostly of Materka and Curtice elements. This is a clear example of how mixing model types can improve accuracy. For the other components the evolved solution is competitive with a Chalmers-based model, with the former having an overall advantage (lower fitness) due to the improved worst case behavior.

$$\begin{bmatrix} 1&1&3&2&1&2&3 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \end{bmatrix}$$



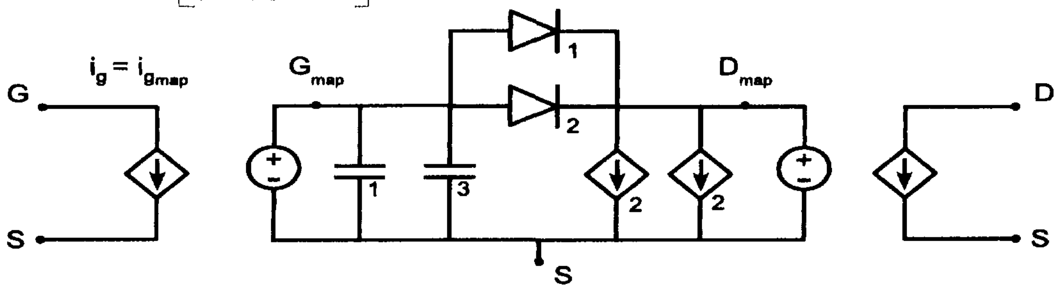
(a) Initial solution from the first generation (fitness = 1.070)

$$\begin{bmatrix} 3&1&0&0&1&2&1 \\ 2&2&2&2&1&0&0 \\ 0&0&0&0&0&0&0 \\ 2&2&2&1&1&0&0 \end{bmatrix}$$



(b) Intermediate solution (fitness = 0.256)

$$\begin{bmatrix} 3&1&0&0&0&1&2 \\ 0&0&0&0&0&0&1 \\ 1&1&2&2&2&0&0 \\ 0&0&1&2&2&0&0 \end{bmatrix}$$



(c) Final solution after 100 generations (fitness = 0.109)

Figure A.17: Genetic algorithm generated solutions for the HEMT example.

A.4 Conclusion and Future Work

We address the problem when existing empirical models have difficulty fitting new devices well. Previous research in this direction has shown that hybrid methods that

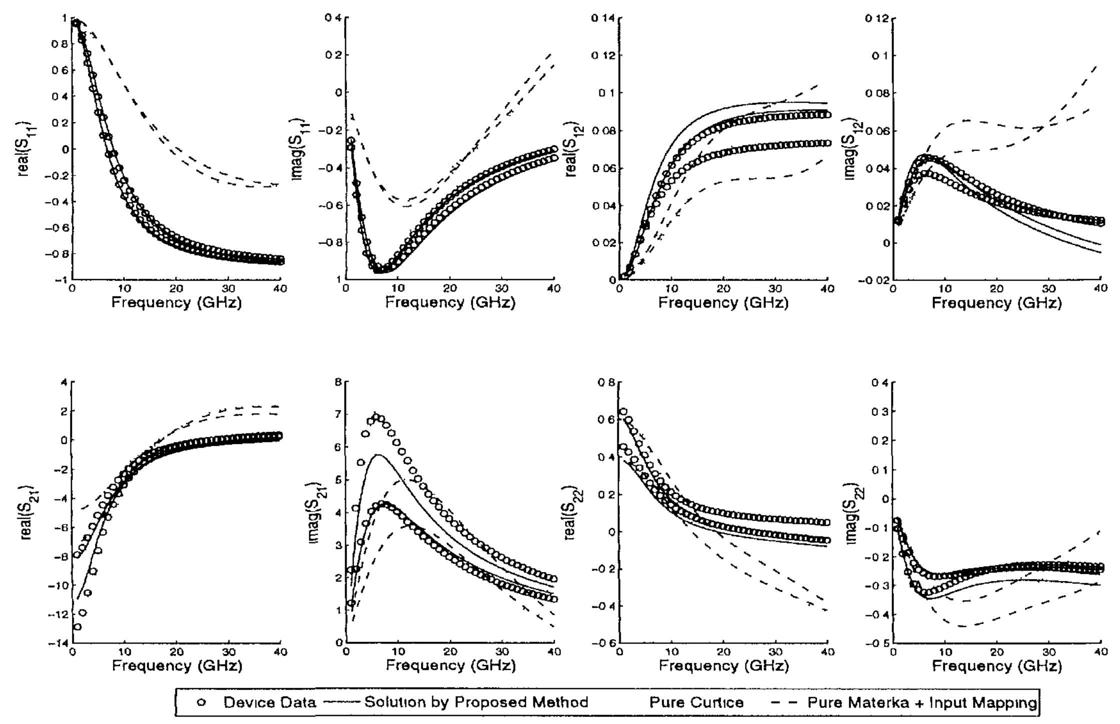


Figure A.18: S-parameter plot for $(V_g, V_d) \in \{(-0.4, 0.1), (-0.2, 3)\}$ *for the HEMT example.*

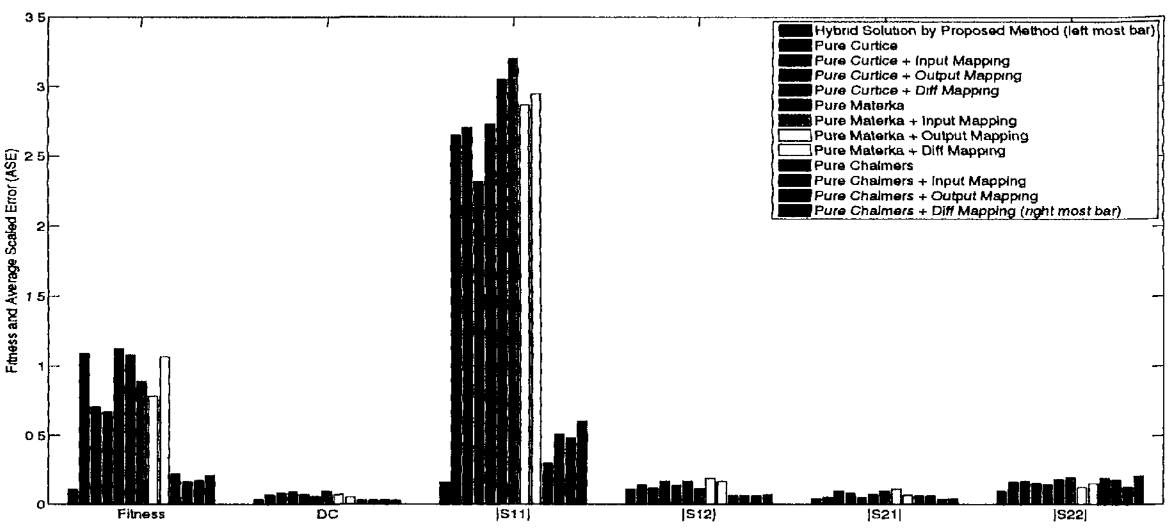


Figure A.19: Comparison of the proposed evolutionary modeling method with pure models on the HEMT modeling problem (Average Scaled Error). A lower fitness value means an overall more accurate model. The solution from the proposed method is a hybrid model with mappings and gives the best accuracy overall.

augment equivalent circuit models with mapping structures can deliver promising results with better generalization. However, determining the optimal combination of mapping structure and equivalent circuit model remains a user intensive process.

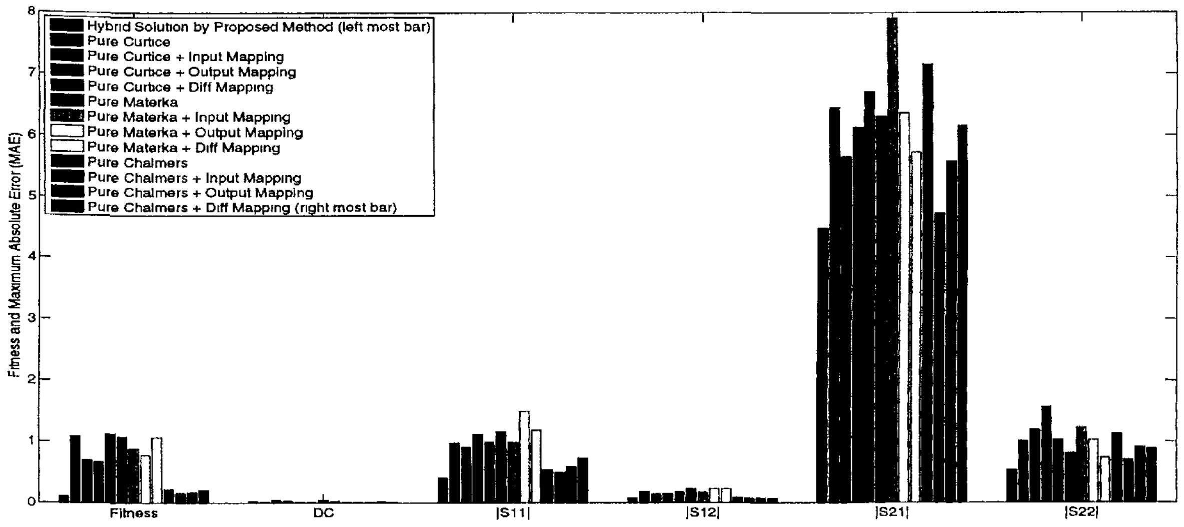


Figure A.20: Comparison of the proposed evolutionary modeling method with pure models on the HEMT modeling problem (Maximum Absolute Error). A lower fitness value means an overall more accurate model. The solution from the proposed method, which is a hybrid model with mappings, gives the best accuracy.

In this paper we have presented an evolutionary approach to knowledge-based modeling of microwave devices that tackles this. Good results were demonstrated on two modeling problems. Through the use of a genetic algorithm, the search for the optimal hybrid combination can be performed more efficiently allowing for potentially large gains in accuracy and performance.

A disadvantage of being based on evolutionary algorithms is that results are not deterministic. However, a strength of this approach is that, since the initial population of the genetic algorithm can be seeded with the existing models, for a particular new device the solution produced by the evolutionary process can be guaranteed to be at least as good as what is currently available. Though in the majority of cases, the generated solution will be superior. A second advantage of evolutionary search is that it runs fully autonomously. Its applicability is limited only by the available computing power (which has currently been commoditized due to the rise of multi-core CPUs, clusters, grids and clouds). Our algorithm has taken advantage of this feature and thus naturally scales with the increasing computing power that is made available.

Naturally room remains for many extensions. The first is to extend the base circuit model to a more general structure containing more nodes. This would allow for more accurate results for more complicated devices. A second natural extension is to extend the possible mapping structures to also include more general ANN-based or other types of mappings. Specifically the Neuro-SM approach described in [230] is of interest here. Further improvements can also be made on the genetic algorithm level itself. Such as the introduction of niching, the refining of the fitness function (e.g., add a penalty proportional to the circuit complexity in order to obtain parsimonious mod-

		Average Scaled Error (ASE)				
	Fitness	**DC**	**|S11|**	**|S12|**	**|S21|**	**|S22|**
Solution of Proposed Method	**0.1091**	0.0361	**0.1598**	0.1096	0.0411	**0.0953**
Pure Curtice	1.0881	0.0687	2.6477	0.1404	0.0537	0.1623
Pure Curtice + Input Mapping	0.7072	0.0817	2.7057	0.1213	0.0949	0.1680
Pure Curtice + Output Mapping	0.6713	0.0902	2.3196	0.1671	0.0833	0.1536
Pure Curtice + Diff. Mapping	1.1234	0.0751	2.7313	0.1392	0.0483	0.1477
Pure Materka	1.0802	0.0567	3.0519	0.1661	0.0739	0.1826
Pure Materka + Input Mapping	0.8916	0.0964	3.2018	0.1152	0.0996	0.1954
Pure Materka + Output Mapping	0.7852	0.0725	2.8681	0.1874	0.1126	0.1267
Pure Materka + Diff. Mapping	1.0669	0.0526	2.9479	0.1657	0.0703	0.1513
Pure Chalmers	0.2235	0.0307	0.3010	0.0670	0.0645	0.1901
Pure Chalmers + Input Mapping	0.1625	**0.0301**	0.5088	0.0669	0.0619	0.1777
Pure Chalmers + Output Mapping	0.1745	0.0341	0.4826	**0.0633**	**0.0384**	0.1278
Pure Chalmers + Diff. Mapping	0.2080	0.0312	0.6037	0.0708	0.0406	0.2072

		Maximum Absolute Error (MAE)				
	Fitness	**DC**	**|S11|**	**|S12|**	**|S21|**	**|S22|**
Solution of Proposed Method	**0.1091**	0.0268	**0.4114**	0.0773	**4.4820**	**0.5441**
Pure Curtice	1.0881	0.0149	0.9737	0.1822	6.4553	1.0153
Pure Curtice + Input Mapping	0.7072	0.0423	0.9116	0.1483	5.6536	1.1947
Pure Curtice + Output Mapping	0.6713	0.0337	1.1196	0.1568	6.1296	1.5647
Pure Curtice + Diff. Mapping	1.1234	0.0149	0.9951	0.1763	6.7160	1.0309
Pure Materka	1.0802	0.0105	1.1604	0.2323	6.3137	0.8150
Pure Materka + Input Mapping	0.8916	0.0412	0.9891	0.1696	7.9045	1.2219
Pure Materka + Output Mapping	0.7852	0.0282	1.4954	0.2308	6.3656	1.0266
Pure Materka + Diff. Mapping	1.0669	0.0123	1.1845	0.2276	5.7310	0.7399
Pure Chalmers	0.2235	**0.0089**	0.5436	0.0907	7.1590	1.1284
Pure Chalmers + Input Mapping	0.1625	0.0182	0.4981	0.0816	4.7307	0.7111
Pure Chalmers + Output Mapping	0.1745	0.0226	0.5986	0.0817	5.5792	0.9093
Pure Chalmers + Diff. Mapping	0.2080	0.0145	0.7365	**0.0748**	6.1568	0.8880

Table A.3: Comparison of the proposed evolutionary modeling method with pure models on the HEMT modeling problem. A lower fitness values means an overall more accurate model. The solution from the proposed method is a hybrid model with mappings and gives the best accuracy overall.

els), and the extension to multi-objective evolutionary optimization. The latter would prevent one having to define different weighting factors for each of the objectives and allow one to tackle the problem directly. However, the computational cost would be much higher.

Finally, one could also consider allowing the genetic algorithm to evolve the empirical formulae themselves. In this case the building blocks are not the formula blocks for each element but the individual equations themselves. This would most likely require a switch to a hierarchical representation and more hardware in order to keep the solution method tractable.

Acknowledgments

This work was supported in part by a grant from the Fund for Scientific Research (FWO), Flanders and in part by the Natural Sciences and Engineering Research council of Canada.

B SUMO Toolbox Developement Information

Computers are useless. They can only give you answers.
- Pablo Picasso

B.1 Introduction

The purpose of this appendix is just to give a brief overview of the development of the SUMO Toolbox. As mentioned in section 4.2.1, the SUMO Toolbox grew out of the M3 Toolbox, an old screenshot of which is shown in figure B.1. The M3 Toolbox was developed for two years (2006-2007) and in April 2008 the first version of the SUMO Toolbox was released. Thus at the time of writing the SUMO software stack has seen 4 years of development by 3 full time researchers in the past three years.

B.2 Infrastructure

To coordinate development the project was initially supported by a standalone server and GForge installation (an old fork of the well known SourceForge project hosting system). This included mailing list (Mailman), revision control (Subversion), Wiki (MediaWiki), forum, and bug tracking functionality. However, this solution turned out to be overkill and difficult to maintain given the relatively small GForge developer community. Thus a transition was made to a more loosely coupled system. The

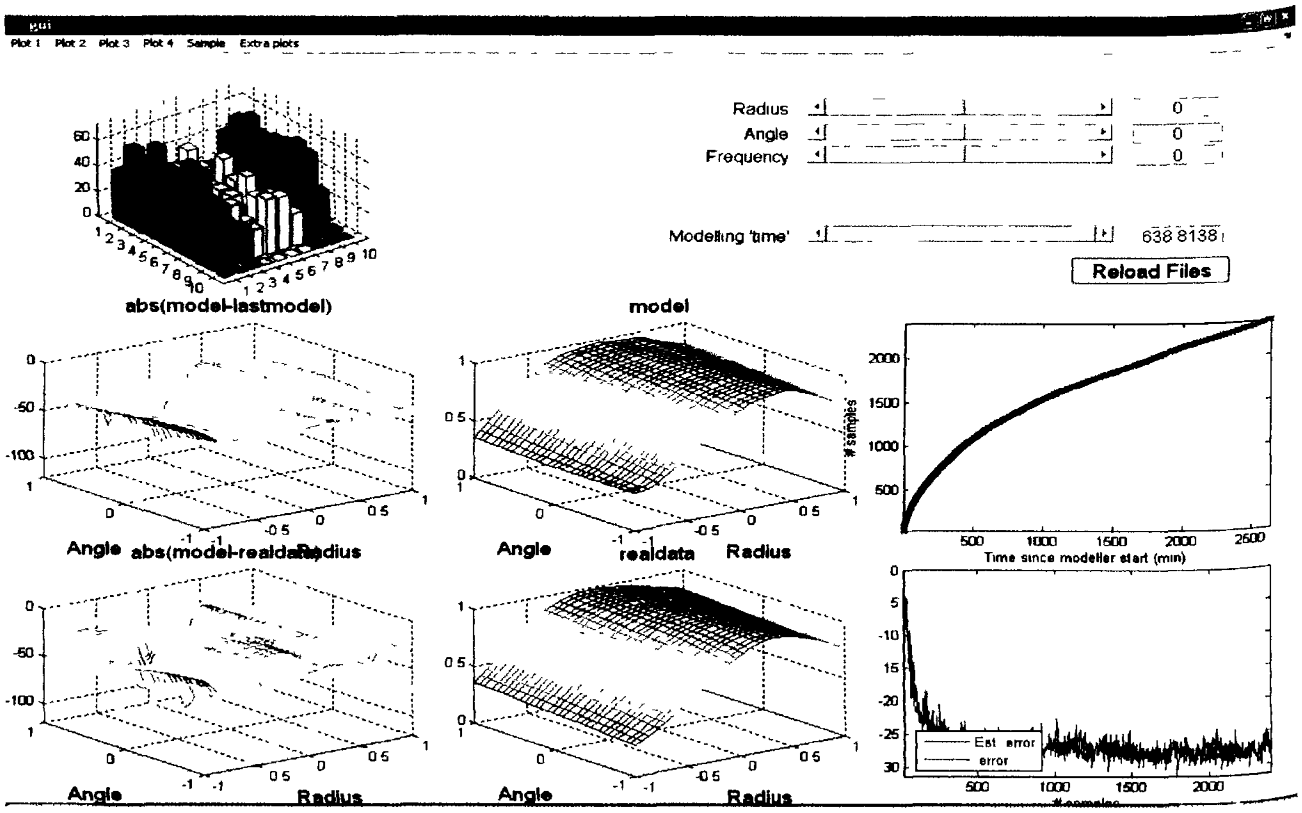


Figure B.1: An old screenshot of one of the first M3-Toolbox versions, highly customized to EM modeling problems.

server (Debian Etch) was configured with a standalone MediaWiki installation and Subversion server and scripts were put in place to enable nightly builds of the latest development snapshots. This would serve as the main site for the SUMO Toolbox: `http://www.sumowiki.intec.ugent.be`. In addition a Drupal installation was setup on the same server to act as portal to the wider res arch group and their activities: `http://www.sumo.intec.ugent.be`. In 2009 this was followed by the start of a blog covering research activities related to the toolbox (`http://sumolab.blogspot.com`) and the start of a YouTube video channel: `http://www.youtube.com/user/sumolab`.

Finally, in addition to this infrastructure, a test server was also setup that would run through the nightly test suite to ensure regressions introduced during development were quickly detected.

B.3 Development statistics

The figures in this section show some development statistics. Note these figures only take into account the Matlab and Java source files and not any of the other files (txt, xml, doc, ppt, etc.) present in the repository. Figure B.2 shows the evolution of the lines of code throughout the project while figure B.3 shows the number of files. Figures B.4 and B.5 then illustrate how this translates to developer activity.

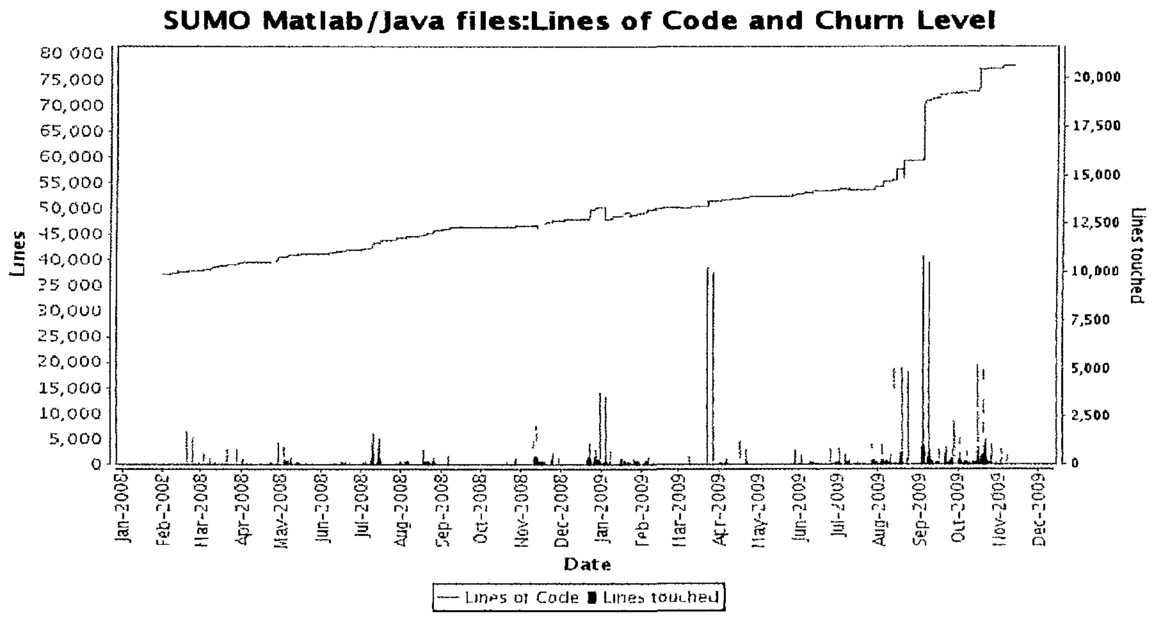


Figure B.2: LOC and Churn shows the usual LOC with the amount of code touched per day. Hopefully this should go decreasingly towards a release.

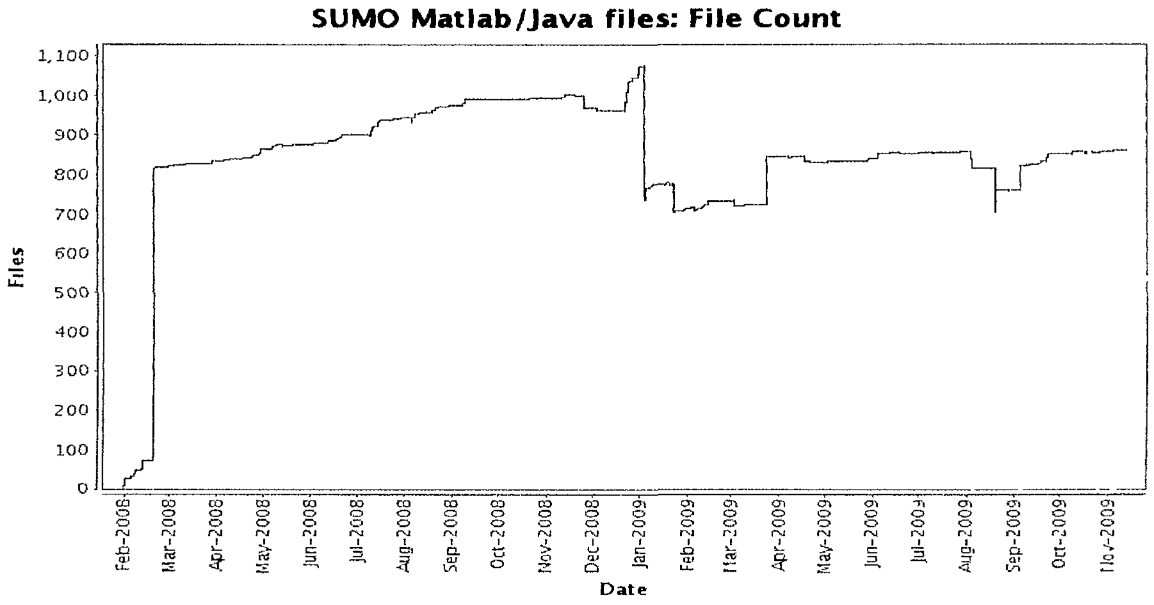


Figure B.3: Number of files in the repository

At the time of writing there have been 6400 commits to the repository with the

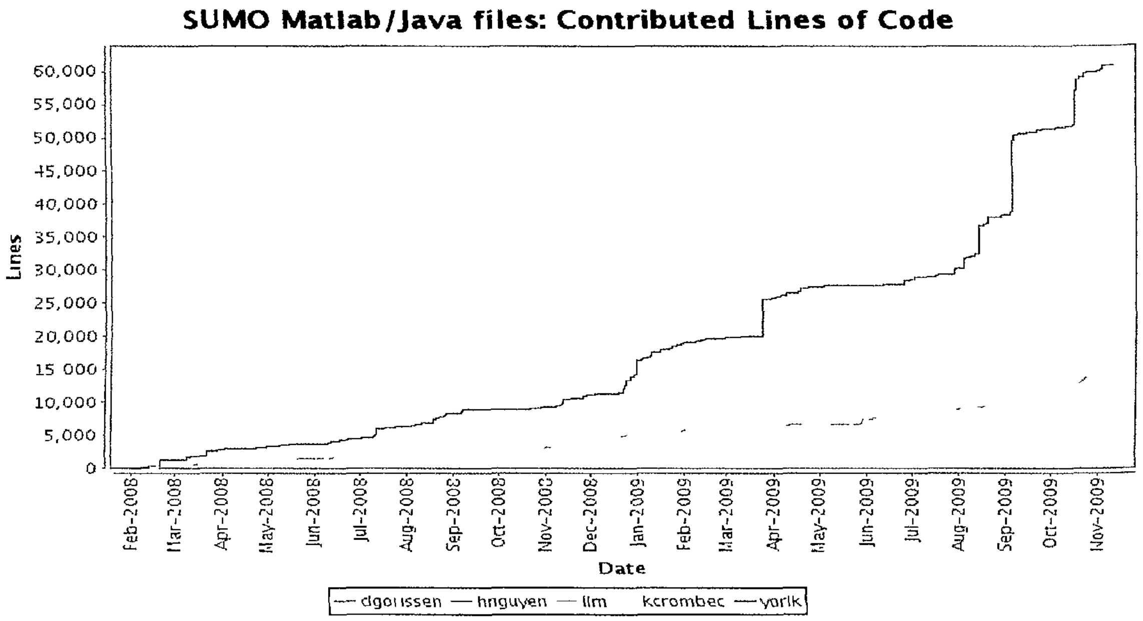


Figure B.4: Contributed LOC per author.

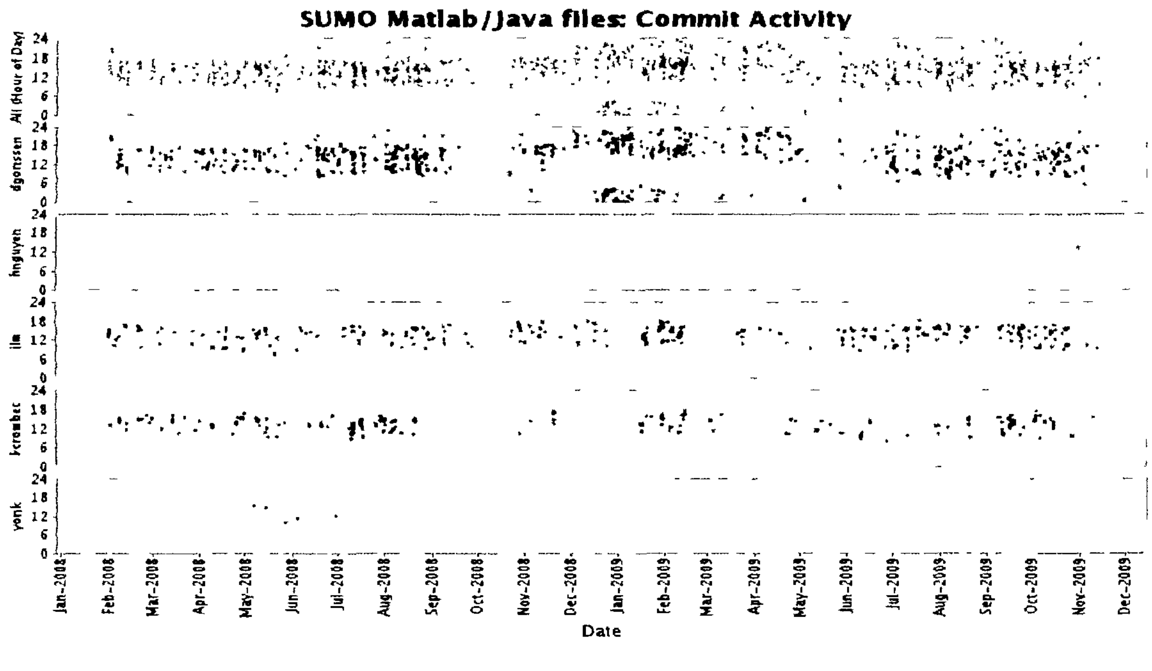


Figure B.5: Commit activity for each author

most popular word in the commit messages being "fixed" (10.5%). A very (!) rough COCOMO II calculation shows these development statistics translate into a software development cost of about 400,000$.

B.4 Downloads

A difficult part of writing software and making it available is estimating how many users are actually using the software and for what means. Users typically only give feedback if something is not working and even then they may not wish to give details on what exactly they are using the software for. Since for most of the lifetime of the project the license was fairly restricted and the download procedure very controlled we have at least a good idea of how many people at least took enough interest in the toolbox to go through the download process. The downside of this is, of course, that many potential users will have been put off by the license or access procedure. A summary of the download statistics of the SUMO Toolbox is shown in table B.1.

Version	Release Date	Days released	Total downloads	Downloads per day
5	04/08/2008	120.00	92	0.767
6	08/06/2008	17.00	73	4.294
6.0.1	08/23/2008	177.00	*N/A*	*N/A*
6.1	02/16/2009	60.00	42	0.700
6.1.1	04/17/2009	172.00	108	0.628
6.2	10/06/2009	13.00	8	0.615
6.2.1	10/19/2009	44.00	42	0.955

Table B.1: SUMO Toolbox download summary as of 2 December 2009.

It is also important to remark that it has really only been since the past one and a half years that the toolbox has been in a stable and accessible enough state, and with adequate documentation to allow more casual experimentation and testing. Older versions would have required a much more specialist user with much higher perseverance.

At the time of writing, about 250-300 users have registered to download the software, of which a small percentage will probably ever have used the toolbox for more than half an hour. However, such things are hard to find out since users do not always respond to questions. A trend does seem though, that while the number of download requests has slowed the past half year, the retention rate seems to have increased. Another trend is the growing proportion of users from Indian and Chinese origins. It will be interesting to see how the use evolves as we move towards an open source license model with a more open download and usage policy.

B.5 Conclusion

This dissertation has of course focused on the research side of things. However, with the development of a publicly available toolbox much time was invested in necessary activities to ensure the software was publicly available and relatively stable. This included: setting up and administering the necessary hardware and software, writing and updating of documentation, setting up test and nightly build infrastructure, keeping track of bugs, fixing release schedules and milestones, user support, etc.

While these required a substantial amount of time and did not directly contribute to the research in short term, these activities did prove invaluable to ensuring long term stability and viability of the research goals. Furthermore, they greatly facilitated the dissemination of research results and proved an excellent means to get in touch with and setup collaborations with the wider research community.

B.6 Version overview

The development changelog across the different versions can be found in `http://sumowiki.intec.ugent.be/index.php/Changelog`.

Bibliography

[1] J. T. Oden, "Revolutionizing engineering science through simulation," National Science Foundation (NSF), Blue Ribbon Panel on Simulation-Based Engineering Science, Tech. Rep., 2006.

[2] G. Wang and S. Shan, "Review of metamodeling techniques in support of engineering design optimization," *Journal of Mechanical Design*, vol. 129, no. 4, pp. 370–380, 2007.

[3] L. Gu, "A comparison of polynomial based regression models in vehicle safety analysis," in *2001 ASME Design Automation Conference, ASME, Pittsburgh, PA*, A. Diaz, Ed., 2001.

[4] A. J. Keane and P. B. Nair, *Computational approaches for Aerospace Design, The Pursuit of Excellence.* The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England: John Wiley & Sons Ltd., 2005.

[5] M. Meckesheimer, "A framework for metamodel-based design: Subsystem metamodel assessment and implementation issues," Ph.D. dissertation, The Pennsylvania State University, 2001.

[6] J. Sacks, W. J. Welch, T. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical science*, vol. 4, no. 4, pp. 409–435, 1989.

[7] J. C. Helton, "Conceptual and computational basis for the quantification of margins and uncertainty," Sandia National Laboratories, Tech. Rep. SAND2009-3055, June 2009.

[8] J. Kleijnen, *Design and Analysis of Simulation Experiments.* Springer, 2008.

[9] Y. Lin, "An efficient robust concept exploration method and sequential exploratory experimental design," Ph.D. dissertation, Georgia Institute of Technology, 2004.

[10] T. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen, "Metamodels for computer-based engineering design: Survey and recommendations." *Eng. Comput. (Lond.)*, vol. 17, no. 2, pp. 129–150, 2001.

[11] R. Jin, W. Chen, and T. Simpson, "Comparative studies of metamodelling techniques under multiple modelling criteria," *Structural and Multidisciplinary Optimization*, vol. 23, no. 1, pp. 1–13, December 2001.

[12] N. Queipo, R. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Tucker, "Surrogate-based analysis and optimization," *Progress in Aerospace Sciences*, vol. 41, pp. 1–28, 2005.

[13] R. Yang, N. Wang, C. H. Tho, and J. P. Bobinaeu, "Metamodeling development for vehicle frontal impact simulation," *Journal of Mechanical Design*, vol. 127, no. 5, pp. 1014–1020, September 2005.

[14] V. Chen, K.-L. Tsui, R. Barton, and M. Meckesheimer, "A review on design, modeling and applications of computer experiments," *IIE Transactions*, vol. 38, pp. 273–291, 2006.

[15] H.-S. Chung and J. J. Alonso, "Comparison of approximation models with merit functions for design optimization," in *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, September 2000, aIAA Paper 2000-4754.

[16] S. Gano, H. Kim, and D. Brown, "Comparison of three surrogate modeling techniques: Datascape, kriging, and second order regression," in *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA-2006-7048, Portsmouth, Virginia,* 2006.

[17] T. Santner, B. Williams, and W. Notz, *The design and analysis of computer experiments*, ser. Springer series in statistics. Springer, 2003.

[18] D. Lim, Y.-S. Ong, Y. Jin, and B. Sendhoff, "A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 07)*. New York, NY, USA: ACM, 2007, pp. 1288–1295.

[19] H. Fang, M. Rais-Rohani, Z. Liu, and M. Horstemeyer, "A comparative study of metamodeling methods for multiobjective crashworthiness optimization," *Computers and Structures*, vol. 83, no. 25-26, pp. 2121–2136, 2005.

[20] D. Gorissen, L. De Tommasi, K. Crombecq, and T. Dhaene, "Sequential modeling of a low noise amplifier with neural networks and active learning," *Neural Computing and Applications*, vol. 18, no. 5, pp. 485–494, Jun. 2009.

[21] D. J. Toal, N. W. Bressloff, and A. J. Keane, "Kriging hyperparameter tuning strategies," *AIAA Journal*, vol. 46, no. 5, pp. 1240–1252, 2008.

[22] D. Busby, C. L. Farmer, and A. Iske, "Hierarchical nonlinear approximation for experimental design and statistical data fitting," *SIAM Journal on Scientific Computing*, vol. 29, no. 1, pp. 49–69, Jan. 2007.

[23] A. Farhang-Mehr and S. Azarm, "Bayesian meta-modelling of engineering design simulations: a sequential approach with adaptation to irregularities in the response behaviour," *International Journal for Numerical Methods in Engineering*, vol. 62, no. 15, pp. 2104–2126, 2005.

[24] V. Devabhaktuni, B. Chattaraj, M. Yagoub, and Q.-J. Zhang, "Advanced microwave modeling framework exploiting automatic model generation, knowledge neural networks, and space mapping," *IEEE Transactions on Microwave Theory and Techniques*, vol. 51, no. 7, pp. 1822–1833, Jul. 2003.

[25] M. Ganser, K. Grossenbacher, M. Schutz, L. Willmes, and T. Back, "Simulation meta-models in the early phases of the product development process," in *Proceedings of Efficient Methods for Robust Design and Optimization (EUROMECH 07)*, 2007.

[26] D. J. Struik, *A Concise History of Mathematics*. Dover Publications, 1987.

[27] H. Schichl, *Models and the history of modeling, in Modeling Languages in Mathematical Optimization*. Kluwer, Boston, 2003, ch. 2, pp. 25–36.

[28] J. Niehans, *A History of Economic Theory*. The John Hopkins University Press, Baltimore, 1990.

[29] S. Hartmann, *Modelling and Simulation in the Social Sciences from the Philosophy of Science Point of View, Theory and Decision Library*. Dordrecht: Kluwer, 1996, ch. The World as a Process: Simulations in the Natural and Social Sciences, pp. 77–100.

[30] R. Frigg and S. Hartmann, "Models in science," in *Stanford Encyclopedia of Philosophy (Spring 2006 Edition)*, 2006.

[31] W. Silvert, "Modeling as a discipline," *Int. J. General Systems*, vol. 30, no. 3, pp. 261–283, 2001.

[32] P. Humphreys, "Numerical experimentation," in *Patrick Suppes Scientific Philosopher*, P. Humphreys, Ed. Dordrecht, 1994, vol. 2, pp. 103–121.

[33] ——, "Computer simulations," in *Proceedings of The Philosophy of Science Association*, A. Fine, M. Forbes, and L. Wessels, Eds., vol. 2. East Lansing, 1991, pp. 497–506.

[34] D. S. Hira and P. K. Gupta, *Operations Research*. Dhanpatrai & Sons, 1999.

[35] G. Gordon, "The development of the general purpose simulation system (gpss)," *ACM SIGPLAN Notices*, vol. 13, no. 8, pp. 183–198, 1978.

[36] J. R. Holmevik, "Compiling simula: A historical study of technological genesis," *IEEE Annals of the History of Computing*, vol. 16, no. 4, pp. 25–37, 1994.

[37] R. Sargent, "Verification and validation of simulation models," in *Winter Simulation Conference, 2005 Proceedings of the*, 4-7 Dec. 2005, p. 14pp.

[38] D. B. Lee, "Requiem for large-scale models," *Journal of the American Institute of Planners*, vol. 39, pp. 163–178, 1973.

[39] W. J. Karplus, "The spectrum of mathematical modeling and systems simulation," *Mathematics and Computers in Simulation*, vol. 19, pp. 3–10, 1977.

[40] ——, "The spectrum of mathematical models," *Perspectives in Computing*, vol. 3, no. 2, pp. 4–13, 1983.

[41] Y.-C. Lin, B. Fregly, R. Haftka, and N. Queipo, "Surrogate-based contact modeling for efficient dynamic simulation with deformable anatomic joints," in *Proceedings of the Tenth International Symposium on Computer Simulation in Biomechanics*, 2005, pp. 23–24.

[42] S. Mohaghegh, "Quantifying uncertainties associated with reservoir simulation studies using surrogate reservoir models," in *SPE Annual Technical Conference and Exhibition (ATCE)*, 2006.

[43] Z. Qian, C. C. Seepersad, J. V. Roshan, J. K. Allen, and C. F. J. Wu, "Building surrogate models based on detailed and approximate simulations," *Journal of Mechanical Design*, vol. 128, no. 4, pp. 668–677, July 2006.

[44] T. W. Simpson, V. Toropov, V. Balabanov, and F. A. C. Viana, "Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come or not," in *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2008 MAO, Victoria, Canada*, 2008.

[45] T. Goodman and R. Spence, "The effect of system response time on interactive computer aided problem solving," in *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1978, pp. 100–104.

[46] T. Simpson, K. Barron, L. Rothrock, M. Frecker, R. Barton, and C. Ligetti, "Impact of response delay and training on user performance with text-based and graphical user interfaces for engineering design," *Research in Engineering Design*, vol. 18, no. 2, pp. 49–65, Aug. 2007. [Online]. Available: http://dx.doi.org/10.1007/s00163-007-0033-y

[47] T. W. Simpson, P. Iyer, K. Barron, L. Rothrock, M. Frecker, R. R. Barton, M., and Meckesheimer, "Metamodel-driven interfaces for engineering design: Impact of delay and problem size on user performance," in *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference and 1st AIAA Multidisciplinary Design Optimization Specialist Conference, Austin, TX, AIAA, AIAA-2005-2060*, 2005.

[48] T. Lenton, R. Marsh, A. R. Price, D. J. Lunt, Y. Aksenov, J. D. Annan, T. Cooper-Chadwick, S. J. Cox, N. R. Edwards, S. Goswami, J. C. Hargreaves, P. P. Harris, Z. Jiao, V. N. Livina, A. J. Payne, I. C. Rutt, J. G. Shepherd, P. J. Valdes, G. Williams, M. S. Williamson, , and A. Yool, "A modular, scalable, grid enabled integrated earth system modelling (genie) framework: Effects of atmospheric dynamics and ocean resolution on bi-stability of the thermohaline circulation." *To appear in Climate Dynamics*, 2007.

[49] R. R. Barton, "Design of experiments for fitting subsystem metamodels," in *WSC '97: Proceedings of the 29th conference on Winter simulation.* New York, NY, USA: ACM Press, 1997, pp. 303–310.

[50] W. Zhao and A. Verbraeck, "A framework for configurable hierarchical simulation in a multiple-user decision support environment," in *Winter Simulation Conference, 2005 Proceedings of the*, 4-7 Dec. 2005, p. 9pp.

[51] J. P. C. Kleijnen, *Handbook of Simulation.* Wiley, New York, 1988, ch. Experimental design for sensitivity analysis, optimization, and validation of simulation models, pp. 173–223.

[52] J. Kleijnen, "Strategic directions in verification, validation and accreditation research: A personal view," in *Proceedings of the 2000 Winter Simulation Conference*, no. urn:nbn:nl:ui:12-83971, 2000, Open Access publications from Tilburg University, pp. 909–916. [Online]. Available: http://ideas.repec.org/p/ner/tilbur/urnnbnnlui12-83971.html

[53] P. Janssen, P. Heuberger, and A. Tiktak, "Metamodelleren bij het mnp-rivm," Milieu- en Natuurplanbureau, Tech. Rep. 550013001/2005, 2005.

[54] G. Weickum, M. Eldred, and K. Maute, "Multi-point extended reduced order modeling for design optimization and uncertainty analysis," in *Proceedings of the 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Rhode Island (paper AIAA-2006-2145)*, May 2006., pp. 1–4.

[55] B. Bond and L. Daniel, "Parameterized model order reduction of nonlinear dynamical systems," *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pp. 487–494, Nov. 2005.

[56] S. Gugercin and A. Antoulas, "A comparative study of 7 algorithms for model reduction," in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 3, 12-15 Dec. 2000, pp. 2367–2372vol.3.

[57] M. Rewienski and J. White, "Model order reduction for nonlinear dynamical systems based on trajectory piecewise-linear approximations," *Linear algebra and its applications*, vol. 415, no. 2–3, pp. 426–454, 2006.

[58] P. K. Davis and J. H. Bigelow, *Motivated Metamodels: Synthesis of Cause-Effect Reasoning and Statistical Metamodeling.* Rand Corporation, The, 2003.

[59] S. S. Ravindran, "Real-time computational algorithm for optimal control of an MHD flow system," *SIAM Journal on Scientific Computing*, vol. 26, no. 4, pp. 1369–1388, 2005.

[60] K. Willcox and A. Megretski, "Fourier series for accurate, stable, reduced-order models in large-scale linear applications," *SIAM Journal on Scientific Computing*, vol. 26, no. 3, pp. 944–962, 2005.

[61] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti, "Recycling Krylov subspaces for sequences of linear systems," *SIAM Journal on Scientific Computing*, vol. 28, no. 5, pp. 1651–1674, 2006.

[62] M. E. Kilmer and E. de Sturler, "Recycling subspace information for diffuse optical tomography," *SIAM Journal on Scientific Computing*, vol. 27, no. 6, pp. 2140–2166, 2006.

[63] M. S. Eldred and D. M. Dunlavy, "Formulations for surrogate-based optimization wiht data fit, multifidelity, and reduced-order models," in *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Protsmouth, Virginia*, 2006.

[64] J. Sterman, *Business dynamics: Systems thinking and modeling for a complex world.* Boston: Irwin/McGraw-Hill., 2000.

[65] O. Nelles, *Non-linear system identification.* Berlin-Heidelberg, Springer Verlag., 2000.

[66] A. Varga, "Model reduction software in the SLICOT library," *Applied and Computational Control, Signals, and Circuits*, vol. 2, pp. 239–282, 2001.

[67] D. P. Solomatine and A. Ostfeld, "Data-driven modelling : some past experiences and new approaches," *Journal of hydroinformatics*, vol. 10, no. 1, pp. 3–22, 2008.

[68] O. C. Lingjaerde and K. Liestol, "Generalized projection pursuit regression," *SIAM Journal on Scientific Computing*, vol. 20, no. 3, pp. 844–857, 1998.

[69] A. Forrester, A. Sobester, and A. Keane, *Engineering Design Via Surrogate Modelling: A Practical Guide.* Wiley, 2008.

[70] I. Enting, T. Wigley, and M. Heimann, "Future emissions and concentrations of carbon dioxide: Key ocean/atmosphere/land analyses," CSIRO Division of Atmospheric Research Commonwealth, Scientific and Industrial Research Organisation, Aspendale, Australia, Tech. Rep. 31, 1994.

[71] G. R. V. Hooss, "A nonlinear impulse response model of the coupled carbon cycle-climate system," *Climate Dynamics*, vol. 18, pp. 189–202, 2001.

[72] P. C. Young, "Data-based mechanistic modelling and validation of rainfall-flow processes," in *M. G. Anderson (Ed.), Model Validation in Hydrological Science.* Chichester: J. Wiley, 2001, pp. 117–161.

[73] J. De Geest, T. Dhaene, N. Faché, and D. De Zutter, "Adaptive CAD-model building algorithm for general planar microwave structures," *IEEE Transactions on Microwave Theory and Techniques*, vol. 47, no. 9, pp. 1801–1809, Sep. 1999.

[74] Q. J. Zhang and K. C. Gupta, *Neural Networks for RF and Microwave Design (Book + Neuromodeler Disk).* Norwood, MA, USA: Artech House, Inc., 2000.

[75] S. Leary, A. Bhaskar, and A. J. Keane, "A knowledge-based approach to response surface modelling in multifidelity optimization," *Journal of Global Optimization*, vol. 26, no. 23, pp. 297–319, 2003.

[76] J. Bandler, Q. Cheng, S. Dakroury, A. Mohamed, M. Bakr, K. Madsen, and J. Sondergaard, "Space mapping: the state of the art," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, no. 1, pp. 337–361, Jan. 2004.

[77] S. Koziel, J. Bandler, and K. Madsen, "Space-mapping-based interpolation for engineering optimization," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 54, no. 6, pp. 2410–2421, June 2006.

[78] J. Zhu, J. Bandler, N. Nikolova, and S. Koziel, "Antenna optimization through space mapping," *Antennas and Propagation, IEEE Transactions on*, vol. 55, no. 3, pp. 651–658, March 2007.

[79] S. Koziel and J. Bandler, "Space-mapping optimization with adaptive surrogate model," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 55, no. 3, pp. 541–547, March 2007.

[80] P. K. Davis and J. H. Bigelow, "Motivated metamodels," in *Proceedings of The 2002 PerMIS Workshop*, 2002.

[81] C. M. Holden and A. J. Keane, "Visualization methodologies in aircraft design," in *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. Reston, USA*, 2004, pp. 1–14.

[82] G. E. P. Box and K. Wilson, "On the experimental attainment of optimum conditions (with discussions)," *Journal of the Royal Statistical Society Series B*, vol. 13, no. 1, pp. 1–45, 1951.

[83] J. P. Kleijnen, "A comment on blanning's metamodel for sensitivity analysis: The regression metamodel in simulation," *Interfaces*, vol. 5, no. 3, pp. 21–23, May 1975.

[84] T. Fabian, J. L. Fisher, M. W. Sasieni, and A. Yardeni, "Purchasing raw material on a fluctuating market," *Operations Research*, vol. 7, pp. 107–122, 1959.

[85] T. Goel, R. Haftka, and W. Shyy, "Comparing error estimation measures for polynomial and kriging approximation of noise-free functions," *Journal of Structural and Multidisciplinary Optimization*, vol. 28, no. 5, pp. 429–442, June 2009.

[86] Y. S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong, *Studies in Fuzziness and Soft Computing Series*. Springer Verlag, 2004, ch. Surrogate-Assisted Evolutionary Optimization Frameworks for High-Fidelity Engineering Design Problems, Knowledge Incorporation in Evolutionary Computation, pp. 307–331.

[87] T. Goel, R. Haftka, W. Shyy, and N. Queipo, "Ensemble of surrogates," *Structural and Multidisciplinary Optimization*, vol. 33, pp. 199–216, 2007.

[88] A. Giunta and M. Eldred, "Implementation of a trust region model management strategy in the DAKOTA optimization toolkit," in *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, 2000.

[89] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, Nov. 1998.

[90] M. J. Sasena, P. Y. Papalambros, and P. Goovaerts, "Metamodeling sampling criteria in a global optimization framework," in *8th AIAA/ USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, AIAA Paper 2000-4921.*, 2000.

[91] I. Parmee, J. Abraham, M. Shackelford, O. F. Rana, and A. Shaikhali, "Towards autonomous evolutionary design systems via grid-based technologies," in *Proceedings of ASCE Computing in Civil Engineering*, Cancun, Mexico, 2005.

[92] D. Abramson, T. Peachey, and A. Lewis, "Model optimization and parameter estimation with Nimrod/O." in *International Conference on Computational Science*, 2006, pp. 720–727.

[93] M. Eldred, D. Outka, C. Fulcher, and W. Bohnhoff, "Optimization of complex mechanics simulations with object-oriented software design," in *Proceedings of the 36th IAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, New Orleans, LA*, 1995, pp. 2406–2415.

[94] A. I. J. Forrester, N. W. Bressloff, and A. J. Keane, "Optimization using surrogate models and partially converged computational fluid dynamics simulations," *Proceedings of the Royal Society*, vol. 462, pp. 2177–2204, 2006.

[95] J. P. C. Kleijnen, W. van Beers, and I. V. Nieuwenhuyse, "Constrained Optimization in Simulation: A Novel Approach," *European Journal of Operational Research*, vol. 202, pp. 164–174, 2010.

[96] J. P. C. Kleijnen, *Advancing the Frontiers of Simulation.* Springer, 2009, ch. Factor screening in simulation experiments: review of sequential bifurcation, pp. 169–173.

[97] H.-Y. Fan, G. S. Dulikravich, and Z.-X. Han, "Aerodynamic data modeling using support vector machines," *Inverse Problems in Science and Engineering*, vol. 13, no. 3, pp. 261–278, June 2005.

[98] G. Dellino, J. P. C. Kleijnen, and C. Meloni, "Robust simulation-optimization using metamodels," in *Proceedings of the 41th Winter Simulation Conference*, 2009.

[99] R. Barton, "Issues in development of simultaneous forward-inverse metamodels," in *Proceedings of the Winter Simulation Conference*, 2005, pp. 209–217.

[100] W. J. H. V. Groenendaal and J. P. C. Kleijnen, "Deterministic versus stochastic sensitivity analysis in investment problems: An environmental case study," *European Journal of Operational Research*, vol. 141, no. 1, pp. 8 – 20, 2002.

[101] T. W. Tewoldeberhan and A. Verbraeck, "Using web services and artificial intelligence techniques to develop simulation models of business networks," in *15th European Simulation Symposium and Exhibition*, 2003.

[102] I. Dahm and J. Ziegler, "Using artificial neural networks to construct a metamodel for the evolution of gait patterns of four-legged walking robots," in *Proceedings of the 5th Conf. on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR2002)*, P. Bidaud and F. B. Amar, Eds. Bury St. Edmunds, UK: Professional Engineering Publ., 2002, pp. 825–832.

[103] S. Xiao, B. Z. Wang, X. Zhong, and G. Wang, “Wideband mobile antenna design based on artificial neural network models,” *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 13, no. 4, pp. 316–320, July 2003.

[104] H. Lee, B. Sanso, W. Zhou, and D. Higdon, “Inference for a proton accelerator using convolution models,” University of Calafornia Santa Cruz, Tech. Rep. ams-2005-31, 2005.

[105] D. Knight, J. Kohn, K. Rasheed, N. Weber, V. Kholodovych, W. Welsh, and J. Smith, “Using surrogate modeling in the prediction of fibrinogen adsorption onto polymer surfaces,” *Journal of chemical information and computer science*, vol. 55, pp. 1088–1097, 2004.

[106] A. Tiktak, J. Boesten, A. V. der Linden, and M. Vanclooster, “Mapping the vulnerability of european groundwater to leaching of pesticides with a process based meta-model of europearl,” *Journal of Environmental Quality*, vol. 35, pp. 1213–1226, 2006.

[107] P. A. Stockwell DRB, “Effects of sample size on accuracy of species distribution models,” *Ecological Modelling*, vol. 148, pp. 1–13, 2002.

[108] A. T. P. Stockwell D., “Comparison of resolution of methods used in mapping biodiversity patterns from point-occurrence data,” *Ecological Indicators*, vol. 3, pp. 213–221, 2003.

[109] S. D. Mohaghegh, A. Modavi, H. Hafez, M. Haajizadeh, M. Kenawy, and S. Guruswamy, “Development of surrogate reservoir models (SRM) for fast track analysis of complex reservoirs,” in *SPE Intelligent Energy Conference and Exhibition. 11-13 April 2006, Amsterdam*, 2006.

[110] C. L. D. Mensink, “A policy oriented model system for the assessment of longterm effects of emission reductions on ozone,” in *25th NATO/CCMS International Technical Meeting on Air Pollution Modelling and its Application, Louvain-la-Neuve, Belgium*, 2001.

[111] S. H. F. G. Margaret Edwards and G. Deffuant, “Comparing an individual-based model of behaviour diffusion with its mean field aggregate approximation,” *Journal of Artificial Societies and Social Simulation*, vol. 6, no. 4, 2003.

[112] D. Hidovic and J. Rowe, “Validating a model of colon colouration using an evolution strategy with adaptive approximations,” in *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO) 2004*, vol. 3103. Springer-Verlag, 2004, pp. 1005–1016.

[113] M. Ding and R. Vemur, “An active learning scheme using support vector machines for analog circuit feasibility classification,” in *18th International Conference on VLSI Design*, Jan. 2005, pp. 528–534.

[114] T. G. Robertazzi and S. C. Schwartz, “An accelerated sequential algorithm for producing D-optimal designs,” *Siam Journal on scientific Computing*, vol. 10, pp. 341–358, Mar. 1989.

[115] A. C. Keys and L. P. Rees, “A sequential-design metamodeling strategy for simulation optimization,” *Computers and Operational Research*, vol. 31, no. 11, pp. 1911–1932, 2004.

[116] D. Tikk, L. T. Kóczy, and T. D. Gedeon, “A survey on universal approximation and its limits in soft computing techniques,” *International Journal of Approximate Reasoning*, vol. 33, no. 2, pp. 185–202, 2003.

[117] V. I. Arnold, “On functions of three variables,” *Doklady Akademii Nauk, USSR*, vol. 114, pp. 679–681, 1957.

[118] A. N. Kolmogorov, “On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition,” *Dokl. Akad. USSR*, vol. 114, pp. 953–956, 1957.

[119] S. Lawrence, C. L. Giles, and A. C. Tsoi, “What size neural network gives optimal generalization? convergence properties of backpropagation (umiacs-tr-96-22),” University of Queensland, St. Lucia, Australia, Tech. Rep., 1996.

[120] M. Davis, R. Sigal, and E. J. Weyuker, *Computability, complexity, and languages: fundamentals of theoretical computer science*. Academic Press, 1994.

[121] M. Li and P. M. Vitnyi, *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 2008.

[122] P. Binev, A. Cohen, W. Dahmen, R. DeVore, and V. Temlyakov, “Universal algorithms for learning theory part I : Piecewise constant functions,” *Journal of Machine Learning Research*, vol. 6, pp. 1297–1321, 2005.

[123] M. C. Kennedy and A. O’Hagan, “Bayesian calibration of computer models (with discussion),” *Journal of the Royal Statistical Society Series B*, vol. 63, pp. 425–464, 2001.

[124] J. Oakley and A. O’Hagan, “Bayesian inference for the uncertainty distribution of computer model outputs,” *Biometrika*, vol. 89, no. 4, pp. 769–784, 2002.

[125] A. O’Hagan, “Bayesian analysis of computer code outputs: a tutorial,” *Reliability Engineering and System Safety*, vol. 91, pp. 1290–1300, 2006.

[126] J. Rougier, "Lightweight emulators for complex multivariate functions, mucm technical report 07/02," University of Durham, Tech. Rep., 2007.

[127] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[128] E. T. Jaynes, *Probability Theory : The Logic of Science*. Cambridge University Press, April 2003.

[129] S. Conti and A. O'Hagan, "Bayesian emulation of complex multi-output and dynamic computer models. research report no. 569/07, submitted to Journal of Statistical Planning and Inference." Department of Probability and Statistics, University of Sheffield, Tech. Rep., 2007.

[130] L. M. J. H. J. Vleeshouwers, "A metamodel for PCLake," RIVM, Bilthoven, Tech. Rep. 703715 007, 2004.

[131] R. A. Fisher, *The Design of Experiments*. Olyver and Boyd Edinburgh, 1935.

[132] R. Myers, *Response surface methodology*. Boston: Allyn and Bacon, Inc., 1971.

[133] R. H. Myers, D. C. Montgomery, and C. Anderson-Cook, *Response surface methodology: process and product optimization using designed experiments*. Wiley, New York., 2009.

[134] J. Fellman, "Gustav Elfving's contribution to the emergence of the optimal experimental design theory," *Statistical Science*, vol. 14, no. 2, pp. 197–200, 1999.

[135] J. Kiefer and J. Wolfowitz, "Optimum designs in regression problems," *Annals of Mathematical Statistics*, vol. 30, pp. 271–294, 1959.

[136] M. Brown, S. Adams, B. Dunlavy, D. Gay, D. Swiler, L. Giunta, A. Hart, W. Watson, J.-P. Eddy, J. Griffin, J. Hough, P. Kolda, T. Martinez-Canales, M. Eldred, and P. Williams, "Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 4.1 users manual," Sandia Labs, Tech. Rep. SAND2006-6337, September 2007.

[137] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[138] J. Kleijnen, S. Sanchez, T. Lucas, and T. Cioppa, "State-of-the-art review: A user's guide to the brave new world of designing simulation experiments," *INFORMS Journal on Computing*, vol. 17, no. 3, pp. 263–289, 2005.

[139] K.-T. Fang, R. Li, and A. Sudjianto, *Design and Modeling for Computer Experiments*. Chapman & Hall/CRC, 2005.

[140] J. R. Koehler and A. B. Owen, "Computer experiments," in *Handbook of Statistics*, S. Ghosh and C. R. Rao, Eds. Elsevier science, 1996, vol. 13, pp. 261–308.

[141] A. Siem, "Property preservation and quality measures in meta-models," Ph.D. dissertation, Tilburg University, Tilburg, Netherlands, 2007.

[142] D. Gorissen, L. De Tommasi, W. Hendrickx, J. Croon, and T. Dhaene, "RF circuit block modeling via kriging surrogates," in *Proceedings of the 17th International Conference on Microwaves, Radar and Wireless Communications (MIKON 2008)*, 2008.

[143] J. P. C. Kleijnen and W. van Beers, "Application-driven sequential designs for simulation experiments: Kriging metamodeling," *Journal of the Operational Research Society*, vol. 55, no. 9, pp. 876–883, Aug. 2004.

[144] R. Gautier and L. Pronzato, "Sequential design and active control," in *Proceedings 1997 AMS-IMS-SIAM Summer Conference, IMS Lectures Notes, New Developments and Applications in Experimental Design*, W. R. N. Flournoy and W. Wong, Eds., vol. 34, 1997, pp. 138–151.

[145] ——, "Active sequential design in nonlinear situations," in *52nd Session of the International Statistics Institute, Helsinki, Finland*, August 1999.

[146] R.Gautier and L. Pronzato, "Adaptive control for sequential design," *Discussiones Mathematicae, Probability & Statistics*, vol. 20, no. 1, pp. 97–114, 2000.

[147] T. Simpson, D. Lin, and W. Chen, "Sampling strategies for computer experiments: Design and analysis," *International Journal of Reliability and Application*, vol. 2, no. 3, pp. 209–240, 2002.

[148] C. J. M. I. C. Turner, "Generic sequential sampling for metamodel approximations," in *DETC'03: ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, Illinois*, 2003.

[149] R. Jin, W. Chen, and A. Sudjianto, "An efficient algorithm for constructing optimal design of computer experiments," *Journal of Statistical Planning and Inference*, August 2003.

[150] ——, "Analytical metamodel-based global sensitivity analysis and uncertainty propagation for robust design, paper 2004010429," in *SAE Transactions, SAE Congress, Detroit*, 2004.

[151] Y. Lin, F. Mistree, J. Allen, and K.-L. Tsui, "Sequential metamodeling in engineering design," in *Proceedings of 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2004.

[152] T. E. Murphy, Y. Lin, K.-L. Tsui, J. K. Allen, V. Chen, and F. Mistree, "Robust engineering design," in *Proceedings of the International Conference on Engineering Design (ICED07), Paris, France*, Jan. 2004.

[153] T. W. Simpson, A. J. Booker, D. Ghosh, A. A. Giunta, P. N. Koch, and R.-J. Yang, "Approximation methods in multidisciplinary analysis and optimization: A panel discussion," *Structural and Multidisciplinary Optimization*, vol. 27, no. 5, pp. 302–313, 2004.

[154] M. J. Sasena, M. Parkinson, M. P. Reed, P. Y. Papalambros, and P. Goovaerts, "Improving an ergonomics testing procedure via approximation-based adaptive experimental design," *Journal of Mechanical Design*, vol. 127, no. 5, pp. 1006–1013, September 2005.

[155] W. Hendrickx and T. Dhaene, "Sequential design and rational metamodelling," in *Proceedings of the 2005 Winter Simulation Conference*, M. Kuhl, S. N. M., F. B. Armstrong, and J. A. Joines, Eds., Dec. 2005, pp. 290–298.

[156] J. P. C. Kleijnen and W. van Beers, "Application-driven sequential designs for simulation experiments: Kriging metamodeling," *Journal of the Operational Research Society*, vol. 55, no. 9, pp. 876–883, Aug. 2004.

[157] C. Turner, R. Crawford, and M. Campbell, "Multidimensional sequential sampling for NURBs-based metamodel development," *Engineering with Computers*, vol. 23, no. 3, pp. 155–174, 2007.

[158] E. Vladislavleva, "Model-based problem solving through symbolic regression via pareto genetic programming," Ph.D. dissertation, Tilburg University, Tilburg, the Netherlands, 2008.

[159] L. Liu, "Could enough samples be more important than better designs for computer experiments?" in *Annual Simulation Symposium*, 2005, pp. 107–115.

[160] R. Lehmensiek, P. Meyer, and M. Muller, "Adaptive sampling applied to multivariate, multiple output rational interpolation models with applications to microwave circuits," *International Journal of RF and microwave computer aided engineering*, vol. 12, no. 4, pp. 332–340, 2002.

[161] D. Deschrijver and T. Dhaene, "Rational modeling of spectral data using orthonormal vector fitting," in *Signal Propagation on Interconnects, 2005. Proceedings. 9th IEEE Workshop on*, 10-13 May 2005, pp. 111–114.

[162] P. Triverio, S. Grivet-Talocia, M. Nakhla, F. G. Canavero, and R. Achar, "Stability, causality, and passivity in electrical interconnect models," *IEEE Transactions on Advanced Packaging*, vol. 30, no. 4, pp. 795–808, 2007.

[163] Y. Xiong, W. . Chen, D. Apley, and X. Ding, "A nonstationary covariance-based kriging method for metamodeling in engineering design," *International Journal for Numerical Methods in Engineering*, vol. 71, no. 6, pp. 733–756, 2007.

[164] D. Wolpert, "The supervised learning no-free-lunch theorems," in *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*, 2001.

[165] D. H. Wolpert, "On the connection between in-sample testing and generalization error," *Complex Systems*, vol. 6, pp. 47–94, 1992.

[166] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, no. 1, pp. 1–58, 1992.

[167] S. Yesilyurt, C. K. Ghaddar, M. E. Cruz, and A. T. Patera, "Bayesian-validated surrogates for noisy computer simulations; application to random media," *SIAM Journal on Scientific Computing*, vol. 17, no. 4, pp. 973–992, 1996.

[168] S. M. Clarke, J. H. Griebsch, and T. Simpson, "Analysis of support vector regression for approximation of complex engineering analyses," in *Proceedings of the 29th Design Automation Conference (ASME Design Engineering Technical Conferences) (DAC/DETC'03)*, Sep 2003.

[169] H. Frohlich and A. Zell, "Efficient parameter selection for support vector machines in classification and regression via model-based global optimization," in *In Proc. of the International Joint Conference on Neural Networks (IJCNN)*, vol. 3, 2005, pp. 1431–1438.

[170] M. Meckesheimer, A. J. Booker, R. Barton, and T. Simpson, "Computationally inexpensive metamodel assessment strategies," *AIAA Journal*, vol. 40, no. 10, pp. 2053–2060, 2002.

[171] D. A. Kenneth P. Burnham, *Model Selection and Multi-Model Inference*. Springer, 2003.

[172] C. Gold, A. Holub, and P. Sollich, "Bayesian approach to feature selection and parameter tuning for support vector machine classifiers," *Neural Networks*, vol. 18, no. 5-6, pp. 693–701, 2005.

[173] D. J. C. MacKay, "Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks," *Network: Computation in Neural Systems*, vol. 6, pp. 469–505, 1995.

[174] C. Perttunen, "Bayesian model parameter estimation of systems subject to random input and output measurement error," in *Systems Engineering, 1989., IEEE International Conference on*, 24-26 Aug. 1989, pp. 227–230.

[175] A. Saksena, D. Lucarelli, and I.-J. Wang, "Bayesian model selection for mining mass spectrometry data." *Neural Networks*, vol. 18, no. 5-6, pp. 843–849, 2005.

[176] D. Madigan, A. Genkin, D. D. Lewis, and D. Fradkin, "Bayesian multinomial logistic regression for author identification," in *AIP Conference Proceedings of 25th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, vol. 803, November 2005, pp. 509–516.

[177] D. Anguita, S. Ridella, F. Rivieccio, and R. Zunino, "Automatic hyperparameter tuning for support vector machines," in *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*. London, UK: Springer-Verlag, 2002, pp. 1345–1350.

[178] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[179] T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi, "General conditions for predictivity in learning theory," *Nature*, vol. 428, pp. 419–422, 2004.

[180] K. Smets, B. Verdonk, and E. M. Jordaan, "Evaluation of performance measures for SVR hyperparameter selection," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN2007)*, 2007.

[181] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *International Joint Conference on Artificial Intelligence*, 1995, pp. 1137–1145.

[182] A. Abd El-Sallam, S. Kayhan, and A. Zoubir, "Bootstrap and backward elimination based approaches for model selection," in *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis*, vol. 1, Sep. 2003, pp. 152–157.

[183] Y.-Y. Ou, C.-Y. Chen, S.-C. Hwang, and Y.-J. Oyang, "Expediting model selection for support vector machines based on data reduction," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 1, 5-8 Oct. 2003, pp. 786–791vol.1.

[184] G. Cawley, "Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs," in *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, 16-21 July 2006, pp. 1661–1668.

[185] J. Kleijnen and R. G. Sargent, "A methodology for the fitting and validation of metamodels in simulation," *European Journal of Operational Research*, vol. 120, pp. 14–29, 2000.

[186] R. Easterling and J. Berger, "Statistical foundations for the validation of computer models," in *Proceedings of the Workshop on Foundations for V&V in the 21st Century*, D. Pace and S. Stevenson, Eds. Society for Modeling and Simulation International, 2002.

[187] M. Berger, J. Paulo, R. Sacks, J. Cafeo, J. Cavendish, J. Lin, C. Bayarri, and J. Tu, "A framework for the validation of computer models." University of Bristol, Statistics Group, Tech. Rep., 2005.

[188] C. Alippi, "Selecting accurate, robust, and minimal feedforward neural networks," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, no. 12, pp. 1799–1810, Dec. 2002.

[189] G. Vazquez Elisa, R. Galindo, P. Joaquin, and Y. Andres, "Model selection methods in multilayer perceptrons," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2, 25-29 July 2004, pp. 1009–1014vol.2.

[190] H. Chen and S. Huang, "A comparative study on model selection and multiple model fusion," in *Information Fusion, 2005 8th International Conference on*, vol. 1, 25-28 July 2005, p. 7pp.

[191] P. L. Bartlett, S. Boucheron, and G. Lugosi, "Model selection and error estimation," *Machine Learning*, vol. 48, no. 1-3, pp. 85–113, 2002.

[192] A. Gelman, "Two-stage regression and multilevel modeling: A commentary," *Political Analysis*, vol. 13, pp. 459–461, 2005.

[193] ——, "Multilevel modeling what it can and cannot do," *Technometrics*, vol. 48, pp. 241–251, 2006.

[194] F. H. Branin, Jr., "Computer methods of network analysis," in *DAC '67: Proceedings of the 4th Design Automation Conference*. New York, NY, USA: ACM, 1967, pp. 8.1–8.19.

[195] G. Kron, *Diakoptics - The Piecewise Solution of Large-Scale Systems*. Macdonald & Company, Ltd., 1963.

[196] G. E. P. Box and R. D. Meyer, "An analysis for unreplicated fractional factorials," *Technometrics*, vol. 28, no. 1, pp. 11–18., 1986.

[197] J. M. Zentner, "A design space exploration process for large scale, multi-objective computer simulations," Ph.D. dissertation, School of Aerospace Engineering, Georgia Institute of Technology, 2006.

[198] H. Hamad and A. Al-Smadi, "Space partitioning in engineering design via metamodel acceptance score distribution," *Engineering with Computers*, vol. 23, no. 3, pp. 175–185, 2007.

[199] S. Yerramareddy and S. C.-Y. Lu, "Hierarchical and interactive decision refinement methodology for engineering design," *Research in Engineering Design*, vol. 4, no. 4, pp. 227–240, 1993.

[200] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of svms for very large scale problems," *Neural Computing*, vol. 14, no. 5, pp. 1105–1114, 2002.

[201] R. B. Gramacy and H. K. H. Lee, "Adaptive design and analysis of supercomputer experiments," *Technometrics*, vol. 51, no. 2, pp. 130–145, May 2009.

[202] A. Sharkey, "On combining artificial neural nets." *Connectionist Science*, vol. 8, no. 3, pp. 299–314, 1996.

[203] L. Hansen and P. Salamon, "Neural network ensembles," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.

[204] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in Neural Information Processing Systems 7, NIPS Conference, Denver, CO*, Dec. 1994, pp. 231–238.

[205] Y. Liu and X. Yao, "A cooperative ensemble learning system," in *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, vol. 3, 4-9 May 1998, pp. 2202–2207vol.3.

[206] L.-W. Chan, "Weighted least square ensemble networks," in *International Joint Conference on Neural Networks*, vol. 2, Jul. 1999, pp. 1393–1396.

[207] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, Nov 2000.

[208] W. Wang, D. Partridge, and J. Etherington, "Hybrid ensembles and coincident-failure diversity," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 4, Jul. 2001, pp. 2376–2381.

[209] Y. Kim, W. Street, and F. Menczer, "Meta-evolutionary ensembles," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 3, May 2002, pp. 2791–2796.

[210] B.-Y. Sun and D.-S. Huang, "Least squares support vector machine ensemble," in *International Joint Conference on Neural Networks*, vol. 3, Jul. 2004, pp. 2013–2016.

[211] G. Webb and Z. Zheng, "Multistrategy ensemble learning: reducing error by combining ensemble learning techniques," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 8, pp. 980–991, Aug. 2004.

[212] Z.-H. Zhou and Y. Jiang, "NeC4.5: neural ensemble based C4.5," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 6, pp. 770–773, June 2004.

[213] Y. Li, R.-P. Yin, Y.-Z. Cai, and X.-M. Xu, "A new decision fusion method in support vector machine ensemble," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol. 6, 18-21 Aug. 2005, pp. 3304–3308Vol.6.

[214] S. Zhou and Z. Sun, "Can ensemble method convert a weak evolutionary algorithm to a strong one?" in *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, vol. 2, 28-30 Nov. 2005, pp. 68–74.

[215] Y. Zhao, J. Gao, and X. Yang, "A survey of neural network ensembles," in *International Conference on Neural Networks and Brain*, vol. 1, Oct. 2005, pp. 438–442.

[216] S. Zhou and R. Chellappa, "From sample similarity to ensemble similarity: probabilistic distance measures in reproducing kernel hilbert space," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 6, pp. 917–929, June 2006.

[217] J. D. Wichard, "Model selection in an ensemble framework," in *Proceedings of International Joint Conference on Neural Networks, Vancouver, Canada*, 2006.

[218] C. Atkeson, A. Moore, and S. Schaal, "Locally weighted learning," *AI Review*, vol. 11, pp. 11–73, April 1997.

[219] W. S. Cleveland and C. L. Loader, *Smoothing by Local Regression: Principles and Methods*. Springer, New York, 1996, pp. 10–49.

[220] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from non-parametric statistics for real time robot learning," *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, July 2002.

[221] V. Toropov, U. Schramm, A. Sahai, R. Jones, and T. Zeguer, "Design optimization and stochastic analysis based on the moving least squares method," in *Proceedings of the 6th World Congress of Structural and Multidisciplinary Optimization, Rio de Janeiro*, 2005.

[222] S. Klanke, S. Vijayakumar, and S. Schaal, "A library for locally weighted projection regression," *J. Mach. Learn. Res.*, vol. 9, pp. 623–626, 2008.

[223] J. P. Kleijnen and W. v. Beers, "Monotonicity-preserving bootstrapped kriging metamodels for expensive simulations," Tilburg University, Open Access publications from Tilburg University RePEc:dgr:kubcen:200975, 2009. [Online]. Available: http://ideas.repec.org/p/ner/tilbur/urnnbnnlui12-3559630.html

[224] D. Deschrijver, T. Dhaene, and D. De Zutter, "Robust parametric macromodeling using multivariate orthonormal vector fitting," *IEEE Transactions on Microwave Theory and Techniques*, vol. 56, no. 7, pp. 1661–1667, Jul. 2008.

[225] F. Wang and Q.-J. Zhang, "Knowledge-based neural models for microwave design," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 45, no. 12, pp. 2333–2343, Dec. 1997.

[226] D. Echeverria and C. Tong, "Towards multilevel optimization: Spacemapping and manifoldmapping," Lawrence Livermore National Laboratory, UCRL-TR-223288, Tech. Rep., 2006.

[227] P. Hemker and D. Echeverría, "A trust-region strategy for manifold-mapping optimization," *Computational Physics*, vol. 224, no. 1, pp. 464–475, 2007.

[228] P. Watson, K. Gupta, and R. Mahajan, "Application of knowledge-based artificial neural network modeling to microwave components," *International Journal of RF and Microwave CAE*, vol. 9, no. 3, pp. 254–260, May 1999.

[229] J. W. Bandler, R. M. Biernacki, S. H. Chen, P. A. Grobelny, and R. H. Hemmers, "Space mapping technique for electromagnetic optimization," *IEEE Transactions on Microwave Theory and Techniques*, vol. 42, no. 12, pp. 2536–2544, Aug. 1994.

[230] L. Zhang, J. Xu, M. C. E. Yagoub, R. Ding, and Q. J. Zhang, "Efficient analytical formulation and sensitivity analysis of neuro-space mapping for nonlinear microwave device modeling," *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, no. 9, pp. 2752–2767, Sep. 2005.

[231] J. Bandler, M. Ismail, J. Rayas-Sanchez, and Q. Zhang, "Neuromodeling of microwave circuits exploiting space mapping technology," in *Proc. IEEE MTT-S International Microwave Symposium Digest*, vol. 1, 1999, pp. 149–152.

[232] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[233] A. J. Booker, J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset, "A rigorous framework for optimization of expensive functions by surrogate," *Structural and Multidisciplinary Optimization*, vol. 17, no. 1, pp. 1–13, 1999.

[234] L. Zhang, Y. Cao, S. Wan, H. Kabir, and Q.-J. Zhang, "Parallel automatic model generation technique for microwave modeling," in *IEEE/MTT-S International Microwave Symposium*, Jun. 2007, pp. 103–106.

[235] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley & Sons, August 1996.

[236] C. W. Krueger, "Software reuse," *ACM Computing Survey*, vol. 24, no. 2, pp. 131–183, 1992.

[237] I. Sommerville, *Software Engineering*, 8th ed. Addison Wesley, 2006.

[238] S. Wagner and F. Deissenboeck, "Abstractness, specificity, and complexity in software design," in *ROA '08: Proceedings of the 2nd international workshop on The role of abstraction in software engineering*. New York, NY, USA: ACM, 2008, pp. 35–42.

[239] K. Crombecq, D. Gorissen, L. D. Tommasi, and T. Dhaene, "A novel sequential design strategy for global surrogate modeling," in *Proceedings of the 41th Conference on Winter Simulation, Austin, TX*, Dec. 2009.

[240] L. Wang, S. Shan, and G. G. Wang, "Mode-pursuing sampling method for global optimization on expensive black-box functions," *Engineering Optimization*, vol. 36, no. 4, pp. 419–438, 2004.

[241] D. Gorissen, I. Couckuyt, and T. Dhaene, "A linear reference model (LRM) metric for model selection and sequential design," University of Antwerp, Tech. Rep., 2009.

[242] K. Ye, W. Li, and A. Sudjianto, "Algorithmic construction of optimal symmetric Latin hypercube designs," *Journal of Statistical Planning and Inference*, vol. 90, pp. 145–159, 2000.

[243] G. Rennen, "Efficient approximation of black-box functions and Pareto sets," Ph.D. dissertation, Tilburg University, Tilburg, The Netherlands, 2009.

[244] B. Husslage, G. Rennen, E. R. van Dam, and D. den Hertog, "Space-filling Latin hypercube designs for computer experiments," Tilburg University, Center for Economic Research, Tech. Rep. 18, 2006.

[245] J.-P. Chiles and P. Delfiner, *Geostatistics: modeling spatial uncertainty.* Wiley, New York, 1999.

[246] H. Wackernagel, *Multivariate geostatistics: an introduction with applications.* Springer-Verlag, Berlin, 2003.

[247] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements od Reusable Object-Oriented Software*, ser. Addison-Wesley Professional Computing Series. New York, NY: Addison-Wesley Publishing Company, 1995.

[248] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," *Towards Global Optimization*, vol. 2, pp. 117–129, 1978.

[249] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Global Optimization*, vol. 21, pp. 345–383, 2001.

[250] M. Sasena, "Flexibility and efficiency enhancements for constrainted global design optimization with kriging approximations," Ph.D. dissertation, University of Michigan, 2002.

[251] D. den Hertog, J. Kleijnen, and A. Siem, "The correct kriging variance estimated by bootstrapping," *Operational Research Society*, vol. 57, pp. 400–409, 2006.

[252] G. Matheron, "Principles of geostatistics," *Economic Geology*, vol. 58, pp. 1246–1266, 1963.

[253] N. A. C. Cressie, *Statistics for spatial data.* John Wiley & Sons, 1993.

[254] M. C. Kennedy and A. O'Hagan, "Predicting the output from complex computer code when fast approximations are available," *Biometrika*, vol. 87, pp. 1–13, 2000.

[255] A. I. Forrester, A. Sobester, and A. J. Keane, "Multi-fidelity optimization via surrogate modelling," *Royal Society*, vol. 463, no. 2088, pp. 3251–3269, 2007.

[256] A. Sóbester, S. J. Leary, and A. J. Keane, "On the design of optimization strategies based on global response surface approximation models," *Global Optimization*, vol. 33, no. 1, pp. 31–59, 2005.

[257] A. Sobester, S. J. Leary, and A. J. Keane, "A parallel updating scheme for approximating and optimizing high fidelity computer simulations," *Structural and Multidisciplinary Optimization*, vol. 27, pp. 371–383, 2004.

[258] W. Ponweiser, T. Wagner, and M. Vincze, "Clustered multiple generalized expected improvement: A novel infill sampling criterion for surrogate models," in *Congress on Evolutionary Computation*, 2008.

[259] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.

[260] A. J. Keane, "Statistical improvement criteria for use in multiobjective design optimization," *AIAA Journal*, vol. 44, no. 4, pp. 879–891, 2006.

[261] I. Voutchkov and A. Keane, "Multiobjective Optimization using Surrogates," in *Adaptive Computing in Design and Manufacture 2006. Proceedings of the Seventh International Conference*, I. Parmee, Ed., Bristol, UK, April 2006, pp. 167–175.

[262] J. Knowles and H. Nakayama, "Meta-modeling in multiobjective optimization," in *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 245–284.

[263] R. Regis and C. Shoemaker, "Constrained global optimization of expensive black box functions using radial basis functions," *Journal of Global Optimization*, vol. 31, no. 1, pp. 153–171, January 2005.

[264] S. Shan and G. G. Wang, "An efficient pareto set identification approach for multi-objective optimization on black-box functions," *Mechanical Design*, vol. 127, no. 5, pp. 866–874, 2005.

[265] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artif. Intell. Rev.*, vol. 18, no. 2, pp. 77–95, 2002.

[266] J. F. M. Barthelemy and R. T. Haftka, "Approximation concepts for optimum structural design a review," *Structural and Multidisciplinary Optimization*, vol. 5, no. 3, pp. 129–144, Sep. 1993.

[267] D. Gorissen, L. De Tommasi, J. Croon, and T. Dhaene, "Automatic model type selection with heterogeneous evolution: An application to RF circuit block modeling," in *Proceedings of the IEEE Congress on Evolutionary Computation, WCCI 2008, Hong Kong*, 2008.

[268] T. Lee, *The Design of CMOS Radio-Frequency Integrated Circuits (Second Edition)*. Cambridge University Press, 2003.

[269] J. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific Publishing Co., Pte, Ltd., 2002.

[270] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard, "Aspects of the matlab toolbox DACE," Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, Tech. Rep., 2002.

[271] D. MacKay, "Bayesian model comparison and backprop nets," in *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. Morgan Kaufmann, Dec. 1992, pp. 839–846.

[272] F. Foresee and M. Hagan, "Gauss-Newton approximation to bayesian regularization," in *Proceedings of the 1997 International Joint Conference on Neural Networks*, Jun. 1997, pp. 1930–1935.

[273] D. Gorissen, "Heterogeneous evolution of surrogate models," Master's thesis, Master of AI, Katholieke Universiteit Leuven (KUL), 2007.

[274] W. Hendrickx, D. Gorissen, and T. Dhaene, "Grid enabled sequential design and adaptive metamodeling," in *WSC '06: Proceedings of the 37th Conference on Winter simulation*. Winter Simulation Conference, 2006, pp. 872–881.

[275] K. Crombecq, I. Couckuyt, E. Laermans, and T. Dhaene, "A novel hybrid active learning strategy for nonlinear regression," in *BeneLearn*, 2009.

[276] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[277] D. Anderson and G. Fedak, "The computational and storage potential of volunteer computing," in *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, vol. 1, 16-19 May 2006, pp. 73–80.

[278] F. Berman, G. Fox, and A. Hey, *Grid Computing: Making The Global Infrastructure a Reality*, Unknown, Ed. John Wiley & Sons, 2003.

[279] K. Hafner and M. Lyon, *Where wizards stay up late : the origins of the Internet / Katie Hafner and Matthew Lyon*. Simon & Schuster, New York :, 1996.

[280] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Lecture Notes in Computer Science*, vol. 2150, 2001.

[281] A. Natrajan, A. Nguyen-Tuong, M. Humphrey, M. Herrick, B. P. Clarke, and A. S. Grimshaw, "The legion grid portal." *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1365–1394, 2002.

[282] M. L. Bote-Lorenzo, Y. A. Dimitriadis, and E. Gómez-Sánchez, "Grid characteristics and uses: A grid definition." in *European Across Grids Conference*, 2003, pp. 291–298.

[283] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE '08*, 2008, pp. 1–10.

[284] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009.

[285] J. P. C. Kleijnen and B. Annink, "Vector computers, Monte Carlo simulation and regression analysis: An introduction," *Management Science*, vol. 38, no. 2, pp. 170–181, 1992.

[286] D. Abramson, A. Lewis, T. Peachey, and C. Fletcher, "An automatic design optimization tool and its application to computational fluid dynamics," in *Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, 2001, pp. 25–25.

[287] R. Buyya and S. Venugopal, "The gridbus toolkit for service oriented grid and utility computing: An overview and status report," in *Proceedings of the First IEEE International Workshop on Grid Economics and Business Models (GECON 2004)*, 2004, pp. 19–36.

[288] M. H. Eres, G. E. Pound, Z. Jiao, J. L. Wason, F. Xu, A. J. Keane, and S. J. Cox, "Implementation and utilisation of a grid-enabled problem solving environment in matlab." *Future Generation Comp. Syst.*, vol. 21, no. 6, pp. 920–929, 2005.

[289] A. Striegel, M. Shorts, E. Stuntebeck, D. Salyers, and J. A. Izaguirre, "GIPSE: A tool for streamling management aspects of the grid for simulation-based research," Univ. of Notre Dame, Tech. Rep., 2004.

[290] M. Yarrow, K. M. McCann, R. Biswas, and R. F. V. der Wijngaart, "An advanced user interface approach for complex parameter study process specification on the information power grid," in *GRID '00: Proceedings of the First IEEE/ACM International Workshop on Grid Computing*. London, UK: Springer-Verlag, 2000, pp. 146–157.

[291] H.-K. Ng, D. Lim, Y.-S. Ong, B.-S. Lee, L. Freund, S. Parvez, and B. Sendhoff, "A multi-cluster grid enabled evolution framework for aerodynamic airfoil design optimization." in *ICNC (2)*, 2005, pp. 1112–1121.

[292] H.-K. Ng, Y.-S. Ong, T. Hung, and B.-S. Lee, "Grid enabled optimization." in *EGC*, 2005, pp. 296–304.

[293] S. Kodiyalam, R. J. Yang, and G. Lei, "Highperformance computing and surrogate modeling for rapid visualization with multidisciplinary optimization," *AIAA journal*, vol. 42, no. 11, pp. 2347–2354, 2004.

[294] W. E. Biles and J. P. C. Kleijnen, "International collaborations in web-based simulation: a focus on experimental design and optimization," in *WSC '05: Proceedings of the 37th conference on Winter simulation.* Winter Simulation Conference, 2005, pp. 218–222.

[295] D. Erwin, "Unicore - a grid computing environment," *Concurrency-Practice and Experience*, vol. 14, pp. 1395–1410, 2002.

[296] E. Laure, S. Fisher, A. Frohner, C. Grandi, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buni, F. Hemmer, A. D. Meglio, and A. Edlund, "Programming the grid using glite," *Computational Methods in Science and Technology*, vol. 12, no. 1, pp. 33–45, 2006.

[297] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing.* Washington, DC, USA: IEEE Computer Society, 2004, pp. 4–10.

[298] W. Gentzsch, "Sun grid engine: Towards creating a compute power grid," in *Proceedings of the 1st International Symposium on Cluster Computing and the Grid.* Washington, DC, USA: IEEE Computer Society, 2001, p. 35.

[299] J. L. Vázquez-Poletti, E. Huedo, R. S. Montero, and I. M. Llorente, "Coordinated harnessing of the IRISGrid and EGEE testbeds with GridWay," *J. Parallel Distrib. Comput.*, vol. 66, no. 5, pp. 763–771, 2006.

[300] M. Massie, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, July 2004.

[301] F. Montesi, C. Guidi, R. Lucchi, and G. Zavattaro, "JOLIE: a java orchestration language interpreter engine," *Electronic Notes in Theoretical Computer Science*, vol. 181, pp. 19 – 33, 2007, combined Proceedings of the Second International

Workshop on Coordination and Organization (CoOrg 2006) and the Second International Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems (MTCoord 2006).

[302] A. J. Keane and P. B. Nair, "Problem solving environments in aerospace design," *Advances in Engineering Software*, vol. 32, no. 6, pp. 477 – 487, 2001.

[303] J. Aernouts, J. Soons, and J. Dirckx, "Quantification of tympanic membrane elasticity parameters from in situ point indentation measurements: Validation and preliminary study," *Hearing Research*, vol. In press, 2009.

[304] R. Dawkins, *The extended phenotype*. Oxford, 1989.

[305] P. Moscato and C. Cotta, "Memetic algorithms," in *Optimization Techniques in Engineering*. Springer-Verlag, 2004, pp. 53–85.

[306] D. Fogel, "Nils Barricelli - artificial life, coevolution, self-adaptation," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 41–45, Feb. 2006.

[307] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989.

[308] S. F. Smith, "A learning system based on genetic adaptive algorithms," Ph.D. dissertation, University of Pittsburgh Pittsburgh, PA, USA, 1980.

[309] N. L. Cramer, "A representation for the adaptive generation of simple sequential programs," in *Proceedings of the 1st International Conference on Genetic Algorithms*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 1985, pp. 183–187.

[310] D. Dickmanns, J. Schmidhuber, and A. Winklhofer, " Der genetische Algorithmus: Eine Implementierung in Prolog. Fortgeschrittenenpraktikum, Institut für Informatik, Lehrstuhl Prof. Radig, Technische Universität München," 1987.

[311] J. R. Koza, *Genetic Programming: On the programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, Mass., 1992.

[312] J. Heitkoetter and D. eds. Beasley, "The Hitch-Hiker's Guide to Evolutionary Computation: A list of Frequently Asked Questions (FAQ), USENET: comp.ai.genetic." Available via anonymous FTP from rtfm.mit.edu/pub/usenet/news.answers/ai-faq/genetic/, 2001.

[313] F. Xiong, Y. Xiong, W. Chen, and S. Yang, "Optimizing Latin hypercube design for sequential sampling of computer experiments," *Engineering Optimization*, vol. 41, no. 8, pp. 793–810, 2009.

[314] J. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press, 1975.

[315] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *Neural Networks, IEEE Transactions on*, vol. 5, no. 1, pp. 96–101, Jan. 1994.

[316] P. Merz and B. Freisleben, "A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3, 6-9 July 1999.

[317] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (3rd ed.).* London, UK: Springer-Verlag, 1996.

[318] R. Poli, "Exact schema theory for genetic programming and variable-length genetic algorithms with one-point crossover," *Genetic Programming and Evolvable Machines*, vol. 2, no. 2, pp. 123–163, 2001.

[319] S. Su and D. Zhan, "New genetic algorithm for the fixed charge transportation problem," in *Proc. of The Sixth World Congress on Intelligent Control and Automation, Dalian, China*, vol. 2, Oct. 2006, pp. 7039–7043.

[320] C.-H. Hsu, "Optimizing beam pattern of adaptive linear phase array antenna using local genetic algorithm," in *Antennas and Propagation Society International Symposium, 2005 IEEE*, vol. 1B, 3-8 July 2005, pp. 315–318vol.1B.

[321] J. Zhang, Y. Zhang, and R. Gao, "Genetic algorithms for optimal design of vehicle suspensions," in *IEEE International Conference on Engineering of Intelligent Systems, Islamabad, Pakistan*, Jan. 2006, pp. 1–6.

[322] N. Siu, E. Elghoneimy, Y. Wang, W. Gruver, M. Fleetwood, and D. Kotak, "Rough mill component scheduling: heuristic search versus genetic algorithms," in *IEEE International Conference on Systems, Man and Cybernetics, The Hague, Netherlands*, vol. 5, Oct. 2004, pp. 4226–4231.

[323] J. Clegg, J. Dawson, S. Porter, and M. Barley, "The use of a genetic algorithm to optimize the functional form of a multi-dimensional polynomial fit to experimental data," in *Proc. IEEE Congress on Evolutionary Computation, Edinburgh, Scotland*, vol. 1, Sep. 2005, pp. 928–934.

[324] R. Tanese, "Distributed genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 434–439.

[325] D. Lim, Y.-S. Ong, Y. Jin, B. Sendhoff, and B.-S. Lee, "Efficient hierarchical parallel genetic algorithms using grid computing," *Future Gener. Comput. Syst.*, vol. 23, no. 4, pp. 658–670, 2007.

[326] M. Nowostawski and R. Poli, “Parallel genetic algorithm taxonomy,” in *Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference*, 31 Aug.-1 Sept. 1999, pp. 88–92.

[327] D. Whitley, S. Rana, and R. B. Heckendorn, “The island model genetic algorithm: On separability, population size and convergence,” *Journal of Computing and Information Technology*, vol. 7, no. 1, pp. 33–47, 1999.

[328] T. Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford, UK: Oxford University Press, 1996.

[329] V. S. Gordon, K. Mathias, and D. Whitley, “Cellular genetic algorithms as function optimizers: locality effects,” in *SAC ’94: Proceedings of the 1994 ACM symposium on Applied computing.* New York, NY, USA: ACM Press, 1994, pp. 237–241.

[330] S. Wright, “The roles of mutation, inbreeding, crossbreeding, and selection in evolution.” in *Proceedings of 6th International Congress on Genetics*, 1932, pp. 356–366.

[331] B. Sareni and L. Krähenbuhl, “Fitness sharing and niching methods revisited.” *IEEE Trans. Evolutionary Computation*, vol. 2, no. 3, pp. 97–106, 1998.

[332] J. Sprave, “Zellulare evolutionare algorithmen zur parameteroptimierung,” in *Informatik in den biowissenschaften.* Springer Informatik aktuell, 1993, pp. 111–120.

[333] K.-J. Kim, J.-O. Yoo, and S.-B. Cho, “Robust inference of bayesian networks using speciated evolution and ensemble.” in *ISMIS*, 2005, pp. 92–101.

[334] K. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[335] M. Moore, “An accurate and efficient parallel genetic algorithm to schedule tasks on a cluster,” in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, 22-26 April 2003, p. 5pp.

[336] K. Belkadi, M. Gourgand, and M. Benyettou, “Parallel genetic algorithms with migration for the hybrid flow shop scheduling problem,” *Journal of Applied Mathematics and Decision Sciences*, vol. 2006, pp. Article ID 65 746, 17 pages, 2006.

[337] K. C. Giannakoglou, M. K. Karakasis, and I. C. Kampolis, “Evolutionary algorithms with surrogate modeling for computationally expensive optimization problem,” in *Proceedings of ERCOFTAC 2006 Design Optimization International Conference, Gran Canaria, Spain*, 2006.

[338] M. Buonanno and D. Mavris, "Aerospace vehicle concept selection using parallel, variable fidelity genetic algorithms," in *Proceedings of 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York*, 2004.

[339] J. Alander, "An indexed bibliography of distributed genetic algorithms, tech. rep. no. 94-1-para," University of Vaasa, Finland, Tech. Rep., 1996.

[340] Z. Konfrst, "Parallel genetic algorithms: advances, computing trends, applications and perspectives," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 26-30 April 2004, p. 162.

[341] G. M. Laslett, "Kriging and splines - an empirical-comparison of their predictive performance in some applications," *Journal of the American Statistical Association*, vol. 89, no. 426, pp. 391–400, 1994.

[342] E. Vladislavleva, G. Smits, and D. den Hertog, "Order of nonlinearity as a complexity measure for models generated by symbolic regression via Pareto genetic programming," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 2, pp. 333–349, April 2009.

[343] M. Streeter and L. Becker, "Automated discovery of numerical approximation formulae via genetic programming," *Genetic Programming and Evolvable Machines*, vol. 4, no. 3, pp. 255–286, 2003.

[344] Y.-S. Yeun, W.-S. Ruy, Y.-S. Yang, and N.-J. Kim, "Implementing linear models in genetic programming." *IEEE Trans. Evolutionary Computation*, vol. 8, no. 6, pp. 542–566, 2004.

[345] P.-W. Chen, J.-Y. Wang, and H.-M. Lee, "Model selection of SVMs using GA approach," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 3, 25-29 July 2004, pp. 2035–2040.

[346] S. Lessmann, R. Stahlbock, and S. Crone, "Genetic algorithms for support vector machine model selection," in *Proceedings of the International Joint Conference on Neural Networks, 2006. IJCNN '06.*, 16-21 July 2006, pp. 3063–3069.

[347] S. Tomioka, S. Nisiyama, and T. Enoto, "Nonlinear least square regression by adaptive domain method with multiple genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 1–16, February 2007.

[348] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple svm parameters." *Neurocomputing*, vol. 64, pp. 107–117, 2005.

[349] C. Zhang, H. Shao, and Y. Li, "Particle swarm optimisation for evolving artificial neural network," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 4, 8-11 Oct. 2000, pp. 2487–2490.

[350] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, Sept. 1999.

[351] X. Yao and Y. Xu, "Recent advances in evolutionary computation." *Journal of Computer Science and Technology*, vol. 21, no. 1, pp. 1–18, 2006.

[352] A. C. Keys, L. P. Rees, and A. G. Greenwood, "Performance measures for selection of metamodels to be used in simulation optimization," *Decision Sciences*, vol. 33, pp. 31 – 58, Oct. 2007.

[353] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions." *IEEE Trans. Evolutionary Computation*, vol. 6, no. 5, pp. 481–494, 2002.

[354] R. Regis and C. Shoemaker, "Local function approximation in evolutionary algorithms for the optimization of costly functions." *IEEE Trans. Evolutionary Computation*, vol. 8, no. 5, pp. 490–505, 2004.

[355] I. Paenke, J. Branke, and Y. Jin, "Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation." *IEEE Trans. Evolutionary Computation*, vol. 10, no. 4, pp. 405–420, 2006.

[356] M. T. M. Emmerich, K. Giannakoglou, and B. Naujoks, "Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels." *IEEE Trans. Evolutionary Computation*, vol. 10, no. 4, pp. 421–439, 2006.

[357] Y.-S. Ong, P. Nair, and K. Lum, "Max-min surrogate-assisted evolutionary algorithm for robust design," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 4, pp. 392–404, Aug. 2006.

[358] E. Sanchez, S. Pintos, and N. Queipo, "Toward an optimal ensemble of kernel-based approximations with engineering applications," in *In Proceedings of the International Joint Conference on Neural Networks, 2006. IJCNN '06.*, 2006, pp. 2152–2158.

[359] Y. B., R. El-Yaniv, and K. Luz, "Online choice of active learning algorithms," *Journal of Machine Learning Research*, vol. 5, pp. 255–291, 2004.

[360] H. J. Escalante, M. Montes, and E. Sucar, "Particle swarm model selection," *Journal of Machine Learning Research (special issue on model selection)*, vol. 10, pp. 405–440, 2008.

[361] H. J. Escalante, M. M. Gomez, and L. E. Sucar, "PSMS for neural networks," in *The IJCNN 2007 Agnostic vs Prior Knowledge Challenge*, 2007, pp. 678–683.

[362] H. Pei and E. Goodman, "A comparison of cohort genetic algorithms with canonical serial and island-model distributed GAs," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001).* San Francisco, California, USA: Morgan Kaufmann, 7-11 July 2001, pp. 501–510.

[363] G. Rawlins, Ed., *Foundations of Genetic Algorithms.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991.

[364] T. Nakama, "Theoretical analysis of genetic algorithms in noisy environments based on a Markov model," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO 08).* New York, NY, USA: ACM, 2008, pp. 1001–1008.

[365] A. Neubauer, "A theoretical analysis of the non-uniform mutation operator for the modified genetic algorithm," in *Proceedings of the IEEE International Conference on Evolutionary Computation,* Apr 1997, pp. 93–96.

[366] X. Qi and F. Palmieri, "Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space. part I: Basic properties of selection and mutation," *IEEE Transactions on Neural Networks,* vol. 5, no. 1, pp. 102–119, Jan 1994.

[367] ——, "Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space. part II: Analysis of the diversification role of crossover," *IEEE Transactions on Neural Networks,* vol. 5, no. 1, pp. 120–129, Jan 1994.

[368] C. A. Ankenbrandt, "An extension to the theory of convergence and a proof of the time complexity of genetic algorithms," in *Foundations of Genetic Algorithms,* 1990, pp. 53–68.

[369] J. P. C. Kleijnen, *Empirical methods for the analysis of optimization algorithms.* Springer, 2009, ch. Design and Analysis of Computational Experiments: Overview.

[370] D. Gorissen, I. Couckuyt, K. Crombecq, and T. Dhaene, "Pareto-based multi-output model type selection," in *Proceedings of the 4th International Conference on Hybrid Artificial Intelligence (HAIS 2009), Salamanca, Spain.* Springer - Lecture Notes in Artificial Intelligence, Vol. LNCS 5572, 2009, pp. 442–449.

[371] I. Couckuyt, D. Gorissen, H. Rouhani, E. Laermans, and T. Dhaene, "Evolutionary regression modeling with active learning: An application to rainfall runoff modeling," in *International Conference on Adaptive and Natural Computing Algorithms,* vol. LNCS 5495, Sep. 2009, pp. 548–558.

[372] G. Smits and M. Kotanchek, "Pareto-front exploitation in symbolic regression," in *Genetic Programming Theory and Practice II*. Springer, Ann Arbor, USA, 2004.

[373] R. Lehmensiek, "Efficient Adaptive Sampling Applied to Multivariate, Multiple Output Rational Interpolation Models, with Applications in Electromagnetics-based Device Modeling," Ph.D. dissertation, University of Stellenbosch, 2001.

[374] S. Rogers, M. Aftosmis, S. Pandya, and N. Chaderjian, "Automated CFD parameter studies on distributed parallel computers," in *Proc of 16th AIAA Computational Fluid Dynamics Conference, Orlando, Florida*, 2003.

[375] R. Gramacy, H. Lee, and W. Macready, "Parameter space exploration with gaussian process trees," in *ICML '04: Proceedings of the 21st International Conference on Machine Learning*. New York, NY, USA: ACM Press, 2004, p. 45.

[376] B. Pamadi, P. Covell, P. Tartabini, and K. Murphy, "Aerodynamic characteristics and glide-back performance of Langley glide-back booster," in *Proceedings of 22nd Applied Aerodynamics Conference and Exhibit, Providence, Rhode Island*, 2004.

[377] D. Harrison and D. Rubinfeld, "Hedonic prices and the demand for clean air," *Journal of Environmental Economics & Management*, vol. 5, pp. 81–102, 1978.

[378] M. Schonlau, "Computer experiments and global optimization," Ph.D. dissertation, University of Waterloo, 1997.

[379] A. I. J. Forrester and D. R. Jones, "Global optimization of deceptive functions with sparse sampling," in *Proceedings of the AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. AIAA, 2008, p. 15.

[380] D. Jones, C. Perttunen, and B. Stuckman, "Lipschitzian optimization without the Lipschitz constant," *Optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181, 1993.

[381] J. Bonnans, J. Gilbert, C. Lemaréchal, and C. Sagastizábal, *Numerical Optimization: Theoretical and Practical Aspects*. Springer, 2006.

[382] Y. Yang and A. Barron, "An asymptotic property of model selection criteria," *Information Theory, IEEE Transactions on*, vol. 44, no. 1, pp. 95–116, Jan 1998.

[383] X. R. Li and Z. Zhao, "Evaluation of estimation algorithms part I: incomprehensive measures of performance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 4, pp. 1340–1358, October 2006.

[384] J. E. Fieldsend, "Multi-objective supervised learning," in *Multiobjective Problem Solving from Nature From Concepts to Applications*, ser. Natural Computing Series, J. Knowles, D. Corne, and K. Deb, Eds. Springer LNCS, 2008.

[385] A. M. Molinaro, R. Simon, and R. M. Pfeiffer, "Prediction error estimation: a comparison of resampling methods," *Bioinformatics*, vol. 21, no. 15, pp. 3301–3307, 2005.

[386] W. Zucchini, "An introduction to model selection," *Journal of Mathematical Psychology*, vol. 44, pp. 41–61, 2000.

[387] B. Efron, "The estimation of prediction error: Covariance penalties and cross-validation," *Journal of the American Statistical Association*, vol. 99, pp. 619–632, January 2004.

[388] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 38, no. 3, pp. 397–415, May 2008.

[389] M. Ihme, A. L. Marsden, and H. Pitsch, "Generation of optimal artificial neural networks using a pattern search algorithm: Application to approximation of chemical systems," *Neural Computation*, vol. 20, pp. 573–601, 2008.

[390] C. R. Rao, "Least squares theory using an estimated dispersion matrix and its application to measurement of signals," in *Proceedings of the Fith Berkeley Symposiu; on Mathematical statistics and Probability*, 1967, pp. 355–372.

[391] Y. Matsuyama, "Harmonic competition: a self-organizing multiple criteria optimization," *IEEE Transactions on Neural Networks*, vol. 7, no. 3, pp. 652–668, May 1996.

[392] F. Fenicia, D. P. Solomatine, H. H. G. Savenije, and P. Matgen, "Soft combination of local models in a multi-objective framework," *Hydrology and Earth System Sciences Discussions*, vol. 4, no. 1, pp. 91–123, 2007.

[393] D. Gorissen, I. Couckuyt, E. Laermans, and T. Dhaene, "Pareto-based multi-output metamodeling with active learning," in *Proceedings of the 11th International Conference on Engineering Applications of Neural Networks (EANN 2009), London, England*, 2009.

[394] I. Mierswa, "Controlling overfitting with multi-objective support vector machines," in *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007, pp. 1830–1837.

[395] T. Suttorp and C. Igel, "Multi-objective optimization of support vector machines," in *Multi-Objective Machine Learning*, 2006, pp. 199–220.

[396] T. Furukawa, C. J. K. Lee, and J. Michopoulos, "Regularization for parameter identification using multi-objective optimization," in *Multi-Objective Machine Learning*, 2006, pp. 125–149.

[397] Y. Jin, "Pareto-based multi-objective machine learning," in *Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on*, 17-19 Sept. 2007, pp. 2–2.

[398] Y. Jin, Ed., *Multi-Objective Machine Learning*, ser. Studies in Computational Intelligence. Springer, 2006, vol. 16.

[399] J. E. Fieldsend and S. Singh, "Pareto multiobjective nonlinear regression modelling to aid CAPM analogous forecasting," in *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, vol. 1, 12-17 May 2002, pp. 388–393.

[400] G. Liu and V. Kadirkamanathan, "Learning with multi-objective criteria," *Fourth International Conference on Artificial Neural Networks*, pp. 53–58, Jun 1995.

[401] J. E. Fieldsend and S. Singh, "Pareto evolutionary neural networks," *Neural Networks, IEEE Transactions on*, vol. 16, no. 2, pp. 338–354, March 2005.

[402] J. S. Armstrong and F. Collopy, "Error measures for generalizing about forecasting methods: Empirical comparisons," *International Journal of Forecasting*, vol. 8, no. 1, pp. 69–80, June 1992.

[403] C. Hennig and M. Kutlukaya, "Some thoughts about the design of loss functions," *REVSTAT - Statistical Journal*, vol. 5, pp. 19–39, 2007.

[404] T. Falas and A.-G. Stafylopatis, "The impact of the error function selection in neural network-based classifiers," *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, vol. 3, pp. 1799–1804 vol.3, 1999.

[405] N. V. Queipo, C. J. Arévalo, and S. A. Pintos, "The integration of design of experiments, surrogate modeling and optimization for thermoscience research," *Engineering with Computers*, vol. 20, no. 4, pp. 309–315, 2005.

[406] A. Price, I. Voutchkov, G. Pound, N. Edwards, T. Lenton, and S. Cox, "Multiobjective tuning of grid-enabled earth system models using a non-dominated sorting genetic algorithm (NSGA-II)," in *e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on*, Dec. 2006, pp. 117–117.

[407] L. Xingtao, L. Qing, Y. Xujing, Z. Weigang, and L. Wei, "Multiobjective optimization for crash safety design of vehicles using stepwise regression model," *Structural and Multidisciplinary Optimization*, vol. 35, no. 6, pp. 561–569, June 2008.

[408] N. M. Sheriffa, N. Guptab, R. Velmuruganc, and N. Shanmugapriyand, "Optimization of thin conical frusta for impact energy absorption," *Thin-Walled Structures*, vol. 46, no. 6, pp. 653–666, 2008.

[409] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[410] K. Deb and P. Nain, "An evolutionary multi-objective adaptive meta-modeling procedure using artificial neural networks." in *Evolutionary Computation in Dynamic and Uncertain Environments*, ser. Studies in Computational Intelligence, S. Yang, Y.-S. Ong, and Y. Jin, Eds. Springer, 2007, vol. 51, pp. 297–322.

[411] K. Vasanth Kumar, K. Porkodi, and F. Rocha, "Comparison of various error functions in predicting the optimum isotherm by linear and non-linear regression analysis for the sorption of basic red 9 by activated carbon," *Journal of hazardous materials*, vol. 150, pp. 158–165, 2008.

[412] A. Mullur and A. Messac, "Metamodeling using extended radial basis functions: a comparative approach," *Eng. with Comput.*, vol. 21, no. 3, pp. 203–217, 2006.

[413] M. Nørgaard, O. Ravn, L. Hansen, and N. Poulsen, "The NNSYSID toolbox," in *IEEE International Symposium on Computer-Aided Control Sysstems Design (CACSD), Dearborn, Michigan, USA*, 1996, pp. 374–379.

[414] K. Sastry, D. Goldberg, and M. Pelikan, "Limits of scalability of multiobjective estimation of distribution algorithms," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 3, 2-5 Sept. 2005, pp. 2217–2224Vol.3.

[415] N. Sangkawelert and N. Chaiyaratana, "Diversity control in a multi-objective genetic algorithm," in *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 4, 8-12 Dec. 2003, pp. 2704–2711Vol.4.

[416] J. Mehnen, T. Wagner, and G. Rudolph, "Evolutionary optimization of dynamic multi-objective test functions," in *Digital Proceedings of the* 3° *Workshop Italiano di Vita Artificiale e della 2a Giornata di Studio Italiana sul Calcolo Evoluzionistico*, S. Cagnoni and L. Vanneschi, Eds. LabSEC – Laboratorio Simulazioni di fenomeni Socio Economici Complessi, September 2006.

[417] G. Agrawal, K. Lewis, K. Chugh, C. Huang, S. Parashar, C. L., and Bloebaum, "Intuitive visualization of pareto frontier for multi-objective optimization in n-dimensional performance space," in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY, AIAA-2004-4434.*, 2004.

[418] M. Jensen, "Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 5, pp. 503–515, Oct. 2003.

[419] Y. Jin, T. Okabe, and B. Sendhoff, "Dynamic Weighted Aggregation for Evolutionary Multi-Objective Optimization: Why Does It Work and How?" in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds. San Francisco, California: Morgan Kaufmann Publishers, 2001, pp. 1042–1049.

[420] A. Schwaighofer and V. Tresp, "Transductive and inductive methods for approximate gaussian process regression," in *NIPS*, 2002, pp. 953–960.

[421] C. Liu and Y. Wang, "Dynamic multi-objective optimization evolutionary algorithm," in *Natural Computation, 2007. ICNC 2007. Third International Conference on*, vol. 4, 24-27 Aug. 2007, pp. 456–459.

[422] I. Hatzakis and D. Wallace, "Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation.* New York, NY, USA: ACM, 2006, pp. 1201–1208.

[423] T. English, "Optimization is easy and learning is hard in the typical function," *Proceedings of the 2000 Congress on Evolutionary Computation, 2000.*, vol. 2, pp. 924–931 vol.2, 2000.

[424] Y. Bengio and N. Chapados, "Extensions to metric based model selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1209–1227, 2003.

[425] C. Cherkassky and F. M. Mulier, *Learning from Data: Concepts, Theory, and Methods.* Wiley-IEEE, 2007.

[426] V. Cherkassky and Y. Ma, "Comparison of model selection for regression," *Neural Comput.*, vol. 15, no. 7, pp. 1691–1714, 2003.

[427] D. Gorissen, I. Couckuyt, E. Laermans, and T. Dhaene, "Multiobjective global surrogate modeling,dealing with the 5-percent problem," *Engineering with Computers*, vol. 26, no. 1, pp. 81–89, Jan. 2010.

[428] J. Hjorth, *Computer Intensive Statistical Methods: Validation, Model Selection, and Bootstrap.* London: Chapman & Hall, 1994.

[429] G. Smits. and E. Vladislavleva, "Ordinal Pareto genetic programming," in *Congress on Evolutionary Computation IEEE*, 16–21 July 2006, pp. 3114–3120.

[430] B.-T. Zhang and H. Mühlenbein, "Balancing accuracy and parsimony in genetic programming," *Evol. Comput.*, vol. 3, no. 1, pp. 17–38, 1995.

[431] T. Soule and J. A. Foster, "Effects of code growth and parsimony pressure on populations in genetic programming," *Evolutionary Computation*, vol. 6, no. 4, pp. 293–309, 1998.

[432] P. L. Bartlett, "For valid generalization the size of the weights is more important than the size of the network," in *NIPS*, 1996, pp. 134–140.

[433] S. Lawrence and C. L. Giles, "Overfitting and neural networks: Conjugate gradient and backpropagation," in *IJCNN (1)*, 2000, pp. 114–119.

[434] R. Caruana, S. Lawrence, and C. L. Giles, "Overfitting in neural networks: Backpropagation, conjugate gradient, and early stopping," in *Advances in Neural Information Processing Systems*, Denver, Colorado, 2001.

[435] P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: risk bounds and structural results," *J. Mach. Learn. Res.*, vol. 3, pp. 463–482, 2003.

[436] P. L. Bartlett, O. Bousquet, and S. Mendelson, "Local Rademacher complexities," *The Annals of Statistics*, vol. 33, no. 4, pp. 1497–1537, 2005.

[437] Z. Chen and S. Haykin, "On different facets of regularization theory," *Neural Computation*, vol. 14, no. 12, pp. 2791–2846, 2002.

[438] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, no. 2, pp. 219–269, 1995.

[439] I. Koo and R. M. Kil, "Model selection for regression with continuous kernel functions using the modulus of continuity," *Journal of Machine Learning Resarch*, vol. 9, pp. 2607–2633, 2008.

[440] G. Lorentz, *Approximation of Functions.* Chelsea Publishing Company, New York, 1986.

[441] O. Hjelle and M. Daehlen, *Triangulations and Applications (Mathematics and Visualization).* Springer, 2006.

[442] G. Bontempi and M. Birattari, "From linearization to lazy learning: A survey of divide-and-conquer techniques for nonlinear control," *International Journal of Computational Cognition*, vol. 3, no. 1, pp. 56–73, 2005.

[443] QHull, "http://www.qhull.org/," 2008.

[444] L. D. Tommasi, J. Rommes, T. Beelen, M. Sevat, J. A. Croon, and T. Dhaene, "Forward and reverse modeling of low noise amplifiers based on circuit simulations," in *Proceedings of the Workshop on Model Reduction for Circuit Simulation, Hamburg, Germany*, 2009.

[445] N. Murata, S. Yoshizawa, and S. Amari, "Network information criterion - determining the number of hidden units for an artificial neural network model," *IEEE Transactions on neural networks*, vol. 5, no. 6, pp. 865–872, 1994.

[446] D. Gorissen, T. Dhaene, and F. DeTurck, "Evolutionary model type selection for global surrogate modeling," *Journal of Machine Learning Research*, vol. 10, pp. 2039–2078, 2009.

[447] J. Shekel, "Test functions for multimodal search techniques," in *In Proceedings of the Fifth Annual Princeton Conference on Information Science and Systems*, 1971.

[448] J. Arnold, R. Srinivasan, R. Muttiah, and J. Williams, "Large area hydrologic modeling and assessment - part 1: Model development," *Journal of the American Water Resources Association*, vol. 34, pp. 73–89, 1998.

[449] E. Bekele and J. Nicklow, "Multi-objective automatic calibration of SWAT using NSGA-II," *Journal of Hydrology*, vol. 341, pp. 165–176, Aug. 2007.

[450] D. A. Savic, G. A. Walters, and J. W. Davidson, "A genetic programming approach to rainfall-runoff modelling," *Water Resources Management*, vol. 13, no. 3, pp. 219–231, 1999.

[451] S. Khu and M. G. F. Werner, "Reduction of Monte Carlo simulation runs for uncertainty estimation in hydrological modelling," *Hydrology and Earth System Sciences*, vol. 7, no. 5, pp. 680–692, 2003.

[452] S. Khu, D. Savic, Y. Liu, and H. Madsen, "A fast evolutionary-based metamodelling approach for the calibration of a rainfall-runoff model," in *Proceedings of the First Biennial Meeting of the International Environmental Modelling and Software Society, Lugano*, 2004.

[453] D. Broad, G. Dandy, H. Maier, and J. Nixon, "Improving metamodel-based optimization of water distribution systems with local search," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 710–717.

[454] M. Kamali, K. Ponnambalam, and E. Soulis, "Computationally efficient calibration of WATCLASS hydrologic models using surrogate optimization," *Hydrology and Earth System Sciences Discussions*, vol. 4, pp. 2307–2321, 2007.

[455] L. Garrote, M. Molina, and L. Mediero, *Practical Hydroinformatics*. Springer Verlag, 2008, vol. 68, ch. Learning Bayesian Networks from Deterministic Rainfall-Runoff Models and Monte Carlo Simulation, pp. 375–388.

[456] H. Rouhani, P. Willems, G. Wyseure, and J. Feyen, "Parameter estimation in semi-distributed hydrological catchment modelling using a multi-criteria objective function," *Hydrological Processes*, vol. 21, no. 22, pp. 2998–3008, 2007.

[457] J. G. Arnold and P. M. Allen, "Automated Methods for Estimating Baseflow and Ground Water Recharge From Streamflow Records," *Journal of the American Water Resources Association*, vol. 35, pp. 411–424, Apr. 1999.

[458] W. Wischmeier and D. Smith, *Predicting Rainfall Erosion Losses. A Guide to Conservation Planning. Agriculture Handbook No. 537.* U.S. Department of Agriculture, USDA, Washington., 1978.

[459] H. Rouhani, D. Gorissen, P. Willems, and J. Feyen, "Improved rainfall-runoff modeling combining a semi-distributed model with artificial neural networks," in *The 4th International SWAT Conference, Delft, The Netherlands*, 2007.

[460] N. Peters, "Laminar diffusion flamelet models in non-premixed turbulent combustion," *Progress in Energy and Combustion Science*, vol. 10, no. 3, pp. 319–339, 1984.

[461] R. Y. Rubinstein, *Simulation and the Monte Carlo Method.* New York, NY, USA: John Wiley & Sons, Inc., 1981.

[462] Z. Wang, "Airfoil geometry design for minimum drag," Purdue University, Tech. Rep. AAE 550, 2005.

[463] UIUC Airfoil Coordinates Database, "http://www.ae.uiuc.edu/m-selig/ads/coord_database.html," 2008.

[464] Design and analysis of subsonic isolated airfoils, "http://web.mit.edu/drela/public/web/xfoil/," 2008.

[465] D. E. Finkel and C. T. Kelley, "Additive scaling and the direct algorithm," *J. of Global Optimization*, vol. 36, no. 4, pp. 597–608, 2006.

[466] F. Aires, C. Prigent, and W. B. Rossow, "Neural network uncertainty assessment using bayesian statistics: A remote sensing application," *Neural Computation*, vol. 16, no. 11, pp. 2415–2458, 2004.

[467] J. Carney, P. Cunningham, and U. Bhagwan, "Confidence and prediction intervals for neural network ensembles," *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, vol. 2, pp. 1215–1218 vol.2, Jul 1999.

[468] A. Olivieri, N. Faber, J. Ferre, R. Boque, J. Kalivas, and H. Mark, "Uncertainty estimation and figures of merit for multivariate calibration," *Pure & Applied Chemistry*, vol. 78, pp. 633–661, 2006.

[469] B. W. Silverman, *Density Estimation for Statistics and Data Analysis.* Chapman & Hall/CRC, Apr. 1986.

[470] D. M. Pozar, *Microwave Engineering*, 2nd ed. John Wiley and Sons, 1998.

[471] K. Crombecq, I. Couckuyt, D. Gorissen, and T. Dhaene, "Space-filling sequential design strategies for adaptive surrogate modelling," in *Soft Computing Technology in Civil, Structural and Environmental Engineering (CSC 2009)*, 2009.

[472] M. Meckesheimer, "A framework for metamodel-based design: Subsystem metamodel assessment and implementation issues," Ph.D. dissertation, The Pennsylvania State University, 2001.

[473] G. Crevecoeur, A. Abdallh, I. Couckuyt, D. Gorissen, L. Depre, and T. Dhaene, "Two-level refined direct method for electromagnetic optimization and inverse problems," in *Proc. of Compumag 2009, Florianopolis, Brasil (Poster session)*, 2009.

[474] I. Couckuyt, K. Crombecq, D. Gorissen, and T. Dhaene, "Automated response surface model generation with sequential design," in *First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering (CSC), Funchal, Portugal*, 2009.

[475] D. Stephens, D. Gorissen, and T. Dhaene, "Surrogate based sensitivity analysis of process equipment," in *Proc. of 7th International Conference on CFD in the Minerals and Process Industries, CSIRO, Melbourne, Australia*, Dec. 2009.

[476] T. Zhu and P. D. Franzon, "Application of surrogate modeling to generate compact and pvt-sensitive ibis models," in *Proceedings of the 18th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, Oct. 2009.

[477] Q. J. Zhang, K. Gupta, and V. Devabhaktuni, "Artificial neural networks for RF and microwave design: from theory to practice," *IEEE Transactions on Microwave Theory and Techniques*, vol. 51, no. 4, pp. 1339–1350, Apr. 2003.

[478] J. E. Rayas-Sanchez, "Em-based optimization of microwave circuits using artificial neural networks: The state-of-the-art," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, no. 1, pp. 420–435, Jan. 2004.

[479] P. Burrascano, S. Fiori, and M. Mongiardo, "A review of artificial neural networks applications in microwave computer-aided design," *International Journal*

of RF and Microwave Computer Aided Engineering, vol. 9, no. 3, pp. 158–174, May 1999.

[480] V. Rizzoli, A. Neri, D. Masotti, and A. Lipparini, "A new family of neural network-based bidirectional and dispersive behavioural models for nonlinear rf/microwave subsystems," *International Journal of RF and Microwave Computer Aided Engineering*, vol. 12, no. 1, pp. 51–70, Jan 2002.

[481] A. H. Zaabab, Q. J. Zhang, and M. Nakhla, "A neural network modeling approach to circuit optimization and statistical design," *IEEE Transactions on Microwave Theory and Techniques*, vol. 43, no. 6, pp. 1349–1358, Jun. 1995.

[482] J. Bandler, N. Georgieva, M. Ismail, J. Rayas-SÃ¡nchez, and J. . Q. J. Zhang vol. 49, pp. 67-79, "A generalized space mapping tableau approach to device modeling," *IEEE Transactions on Microwave Theory and Techniques*, vol. 49, no. 1, pp. 67–79, Jan. 2001.

[483] S. Koziel and J. W. Bandler, "Coarse and surrogate model assessment for engineering design optimization with space mapping," in *Proc. IEEE/MTT-S International Microwave Symposium, Honolulu, HI*, Jun. 2007, pp. 107–110.

[484] ——, "Space mapping with multiple coarse models for optimization of microwave components," *IEEE Microwave and Wireless Components Letters*, vol. 18, no. 1, pp. 1–3, Jan. 2008.

[485] L. Zhang, Q. J. Zhang, and J. Wood, "Statistical neuro-space mapping technique for large-signal modeling of nonlinear devices," *IEEE Transactions on Microwave Theory and Techniques*, vol. 56, no. 11, pp. 2453–2467, Nov. 2008.

[486] M. Steer, J. Bandler, and C. Snowden, "Computer-aided design of RF and microwave circuits and systems," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 3, pp. 996–1005, Mar. 2002.

[487] C. M. Snowden, *Semiconductor Device Modeling*. Peter Peregrinus Ltd., London, UK, 1988.

[488] M. A. Khatibzadeh and R. J. Trew, "A large-signal analytical model for the GaAs MESFET," *IEEE Transactions on Microwave Theory and Techniques*, vol. 36, no. 2, pp. 231–238, Feb. 1988.

[489] K. Lehovec and R. Zuleeg, "Voltagecurrent characteristics of GaAs JFETs in the hot electron range," *Solid State Electron*, vol. 13, pp. 1415–1426, 1970.

[490] C. G. Morton, J. S. Atherton, C. M. Snowden, R. D. Pollard, and M. J. Howes, "A large-signal physical HEMT model," in *Proc. IEEE MTT-S International Microwave Symposium Digest, San Francisco, CA*, vol. 3, Jun. 1996, pp. 1759–1762.

[491] W. R. Curtice, "GaAs MESFET modeling and nonlinear CAD," *IEEE Transactions on Microwave Theory and Techniques*, vol. 36, no. 2, pp. 220–230, Feb. 1988.

[492] H. Statz, P. Newman, I. W. Smith, R. A. Pucel, and H. A. Haus, "GaAs FET device and circuit simulation in SPICE," *IEEE Transactions on Electron Devices*, vol. 34, no. 2, pp. 160–169, Feb. 1987.

[493] A. Materka and T. Kacprzak, "Computer calculation of large-signal GaAs FET amplifier characteristics," *IEEE Transactions on Microwave Theory and Techniques*, vol. 33, no. 2, pp. 129–135, Feb. 1985.

[494] I. Angelov, H. Zirath, and N. Rosman, "A new empirical nonlinear model for HEMT and MESFET devices," *IEEE Transactions on Microwave Theory and Techniques*, vol. 40, no. 12, pp. 2258–2266, Dec. 1992.

[495] V. I. Cojocaru and T. J. Brazil, "A scalable general-purpose model for microwave FETs including DC/AC dispersion effects," *IEEE Transactions on Microwave Theory and Techniques*, vol. 45, no. 12, pp. 2248–2255, Dec. 1997.

[496] C. Snowden, "Nonlinear modelling of power FETs and HBTs," *International Journal of Microwave and Millimeter-wave Computer-Aided Engineering*, vol. 6, no. 4, pp. 219–233, Dec. 1996.

[497] D. E. Root, S. Fan, and J. Meyer, "Technology independent large-signal non quasistatic FET models by direct construction from automatically characterized device data," in *Proc. IEEE 21st European Microwave Conference, Stuttgart, Germany*, Oct. 1991, pp. 927–932.

[498] L. Zhang, J. Xu, M. Yagoub, R. Ding, and Q. Zhang, "Neuro-space mapping technique for nonlinear device modeling and large-signal simulation," in *Proc. IEEE MTT-S International Microwave Symposium Digest, Philadelphia, PA*, Jun. 2003, pp. 173–176.

[499] L. Zhang, R. Ding, and Q. J. Zhang., "Generalized knowledge-based neural network for microwave modeling," in *9th International Symposium On Antenna Technology and Applied Electromagnetics, Montreal, QC*, Aug. 2002, pp. 558–561.

[500] S. Koziel and J. Bandler, "Support-vector-regression-based output space-mapping for microwave device modeling," in *IEEE MTT-S International Microwave Symposium Digest, Atlanta, GA*, Jun. 2008, pp. 615–618.

[501] G. W. Greenwood and A. Tyrrell, *Introduction to Evolvable Hardware: A Practical Guide for Designing Self-Adaptive Systems*. Wiley, 2006.

[502] T. Nishino and T. Itoh, "Evolutionary generation of microwave line-segment circuits by genetic algorithms," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 9, pp. 2048–2055, Sep. 2002.

[503] J. R. Koza, I. Bennett, F. H., D. Andre, M. A. Keane, and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Transactions on Evolutionary Computing*, vol. 1, no. 2, pp. 109–128, Feb. 1997.

[504] X. Yao, "Following the path of evolvable hardware," *Communications of the ACM*, vol. 42, no. 4, pp. 46–49, Apr. 1999.

[505] X. Yao and T. Higuchi, "Promises and challenges of evolvable hardware," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 29, no. 1, pp. 87–97, Jan. 1999.

[506] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, no. 4, pp. 308–313, Jan. 1965.

[507] H. Ninomiya, S. Wan, H. Kabir, X. Zhang, and Q. Zhang, "Robust training of microwave neural network models using combined global/local optimization techniques," in *Proc. IEEE MTT-S International Microwave Symposium Digest, Atlanta, GA*, Jun. 2008, pp. 995–998.

[508] D. Schreurs, M. Verspecht, S. Vandenberghe, and E. Vandamme, "Straightforward and accurate nonlinear device model parameter-estimation method based on vectorial large-signal measurements," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 10, pp. 2315–2319, Oct. 2002.

[509] D. Paul, M. Nakhla, R. Achar, and A. Weisshaar, "Broadband modeling of high-frequency microwave devices," *IEEE Transactions on Microwave Theory and Techniques*, vol. 57, no. 2, pp. 361–373, Feb. 2009.

[510] E. R. Harold and W. S. Means, *XML in a Nutshell.* O'Reilly Media, Inc., Cambridge, Massachusetts, 2004.

[511] P. Walmsley, *XQuery.* O'Reilly Media, Inc., Cambridge, Massachusetts, 2007.

[512] Advanced Design System (ADS) 2006, Agilent Technologies, Palo Alto, CA.

[513] The MathWorks Inc., Natick, MA, USA.

[514] MINIMOS-NT, Institute for Microelectronics, Technical University, Vienna, Austria. Release 2.0.