DISSERTATION

Planning Experiments for the Validation of Electronic Control Units

Systematic Experiment Planning for the Validation
of Automotive Electronic Control Units

Submitted at the Faculty of Electrical Engineering and Information Technology, Vienna University of Technology in partial fulfillment of the requirements for the degree of Doktor der technischen Wissenschaften

under supervision of

Univ.-Prof. Dr.habil. Christoph Grimm
Institut number: 384
Institute of Computer Technology

and

Priv.-Doz. Dr.rer.nat. Georg Pelz
Design Methodologies, Automotive Power
Infineon Technologies

by

Dipl.-Ing. Monica Rafaila
Matrikelnummer: e0727803
Tumblingerstr. 56, 80337 München, Deutschland

27.09.2010 _____

Abstract

Starting from pre-silicon verification, automotive electronic systems require a high degree of reliability, while their functional and structural complexity constantly increases. Despite high-performance simulation-based tools and methods, verification is still too time consuming. One of the reasons is the impact on the target functionality of sources of variability e.g. tolerances of components or to-be-adjusted design parameters, which can come from internal or external influences. A dependable system must comply with requirements even in the presence of such variations, thus they must be accounted for early in the verification flow. Too little focus has been put on planning simulation experiments to extract maximum information about the system reliability when these factors vary.

The objective of this work is to determine the impact multiple sources of variation have on system outputs and find the worst-case behaviour under predefined test scenarios, in order to ensure the system is reliable or find the reasons why it is not. The necessary effort, mainly characterized by the number of simulation runs which is invested, must be reduced, while a large set of variable factors must be effectively handled.

Design of Experiments concepts, completed by metamodelling of the results space and random test methods are analyzed, adapted and extended to perform sensitivity and worst-case analyses. The steps to realize efficient simulation experiments are detailed. Then, a sequential flow is implemented, which replaces the decision-making steps by an adaptive, self correcting process. The worst-case estimate and its associated confidence are then optimized by further iterative gradient-based search. The complete flow is extended for transient analysis, to characterize the system behaviour over the complete simulated time frame. Results of simulation experiments on selected automotive electronic control units reveal the efficiency of the proposed methods, compared to alternatives.

Kurzfassung

Hohe Anforderungen an die Zuverlässigkeit heutiger Automobilelektronik beginnen bereits in der Pre-Silicon-Verifikation, da ihre funktionelle und strukturelle Komplexität sich ständig erhöht. Trotz des Einsatzes leistungsfähiger Simulationsmethoden und Tools ist die Verifikation sehr zeitintensiv. Einer der Gründe hierfür sind die Auswirkungen auf die Funktionalität von Toleranzen in Komponenten oder anzupassenden Designparametern, die aus internen oder externen Einflüssen stammen. Ein zuverlässiges System muss seine Funktionalität auch im Falle solcher Variationen erfüllen. Deshalb müssen sie bereits frühzeitig im Verifikationsverlauf berücksichtigt werden. In der Vergangenheit wurde wenig Fokus auf die Durchführung von Simulations-Experimenten gerichtet, durch die maximale Informationen über die Zuverlässigkeit eines Systems unter solchen variablen Faktoren zu erhalten sind.
Ziel dieser Arbeit ist es, die Auswirkungen mehrerer Variationen auf das System zu untersuchen und das Worst-Case Verhalten unter definierte Testszenarien zu finden, um das Systems dadurch zu validieren oder die Ursachen für den Ausfall zu ermitteln. Der benötigte Aufwand, welcher hauptsächlich von der Anzahl der durchzuführenden Simulationen abhängt, muss optimiert werden. Zusätzlich müssen zahlreiche variable Faktoren effizient behandelt werden.
"Design of Experiments"-Konzepte, welche durch Metamodellierung und Random-Test-Methoden ergänzt werden, werden analysiert, angepasst und erweitert, um Empfindlichkeit- und Worst-Case-Analysen durchzuführen. Die Schritte, die benötigt werden, um effiziente Simulations-Experimente zu erreichen sind detailliert beschrieben. Anschließend wird ein sequentieller Ablauf implementiert, um die Schritte der Entscheidungsfindung durch einen selbstkorrigierenden Prozess zu ersetzen. Die Worst-Case Abschätzung und die damit verbundene Konfidenz werden anschließend mittels einer iterativen gradienten-basierten Suche optimiert. Der gesamte Ablauf wird um eine transiente Analyse erweitert, um das Verhalten des Systems über den gesamten Simulationsverlauf zu charakterisieren. Die Ergebnisse der Simulationsexperimente von ausgewählten elektronischen Kraftfahrzeug-Steuergeräten zeigen die Effizienz der vorgestellten Methoden im Vergleich zu Alternativen.

Acknowledgments

I would like to express my deepest gratitude to Georg Pelz, the leader of our design methodologies group at Infineon. He guided me on this academic path and created the ideal context for my work, challenging, supporting, as well as welcoming. I thank him for introducing me to this research area and sharing valuable knowledge on related topics; for all the motivation, encouragement, appreciation and recognition of work.
To Prof. Christoph Grimm I thank for being supportive along the way, with an active academic implication and interest in my work; for being open to my ideas and encouraging me to follow them; for sharing knowledge and ideas, and patiently guiding me on the path which I chose.
Christian Decker is a valuable advisor. I learned from him that there are no problems, but only challenges, and solutions can always be found if the problem is formulated well enough. He contributed to my personal and professional development.
I appreciate the optimal environment Infineon provided for my work, in the context of the Auto-SUN project[1]. Jerome Kirscher and the rest of the design methodology department were always helpful. Christian Koehler helped me especially with getting my work on paper and I thank him for that.
I thank the group from the Fraunhofer Institute, IIS/EAS, with whom I collaborated mostly in the first part of the PhD. Thomas Markwirth in particular showed full support and contributed to my practical as well as methodology-related skills. To the group at the Institute for Computer Technology of TU Vienna, for being always welcoming in Vienna and responding with interest and useful feedback to my work.
To Prof. Corneliu Burileanu, from the Polytechnic University of Bucharest, the person who introduced me not only to the academic world, but also to my future employer: many thanks for making it possible to pursue this work in this environment.
I am grateful to my parents and my sister for all the love and support. They taught me to appreciate the value of education. I thank them for standing beside me and ensuring that I have the best conditions to follow the choices I make. I hope to make them proud.

[1]This research project is supported by the German Government, Federal Ministry of Education and Research under the grant number 01M3178.

Table of Contents

1 Introduction

Automotive electronic control units (ECUs) need extensive verification. This is a fact now, more than ever, as the verification must be done for complete systems, in the form of SoCs (Systems-on-Chip) or Systems-in-a-Package, integrating what used to be delivered at a component-level. Requirements relate to functionality, computing power, safety, energy efficiency and stability. These all translate into more complexity, both as density of integration and heterogeneity.

Safety is crucial in the automotive sector because any point of potential system failure can have great leverage, endanger costly and time consuming design projects and even the lives of the vehicle occupants. Therefore, the verification flow must ensure as early as possible a high degree of reliability, i.e. a low probability that the system responds outside the admitted range. The next section details the typical structure and verification flow for these systems. Then, a gap in this flow is identified and an overview of how the presented work deals with the problems is provided.

1.1 Automotive electronic control units

An overview of typical automotive systems under verification is presented. The general approach towards pre-silicon development flow in the industrial environment is then described.

1.1.1 Functional and architectural overview

Figure 1.1 shows functional blocks which are present in standard automotive ECUs.

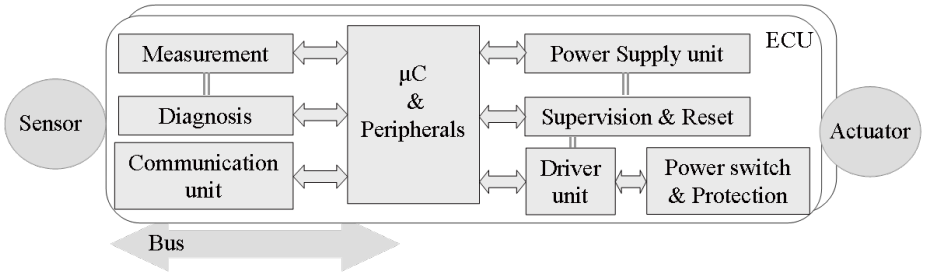


Figure 1.1: ECU functional blocks

During normal operation, a sensor input, converted into electric signals, is processed to produce the control signals directed to the actuators, like valves, relays, lamps, motors [SB04b]. Functions of supply, supervision, communication, conditioning or control are realized in different blocks, in software and hardware (analog and digital). The central digital element is a microcontroller subsystem, which can contain multiple cores if more processing power is needed. The ECU communicates with other control units on a bus subsystem e.g. CAN, through a dedicated transceiver.

Such systems are heterogeneous[1] and introduce sources of variation of multiple natures e.g. calibration parameters of a regulation loop, the tolerance of an electrical component or the fluctuant transmission delay on the bus. For a complete validation, they need to be tested together with the surrounding parts in their operating environment. Hence the need to cover multiple domains: electrical, thermal, mechanical, and handle the interferences between these domains. While the ECU complexity and heterogeneity increase, the time to reach a dependable validation must be decreased. The next section details the typical verification flow.

1.1.2 Industry verification flow

Figure 1.2 shows the well known V-model, applied for design and verification of complex systems and highlights the aspects relevant for the approach. An automotive ECU must be validated as a complete system early in the design process, before going down in the development chain, in order to ensure conformance to requirements e.g. customer expectations, standards' compliance or budget constraints.

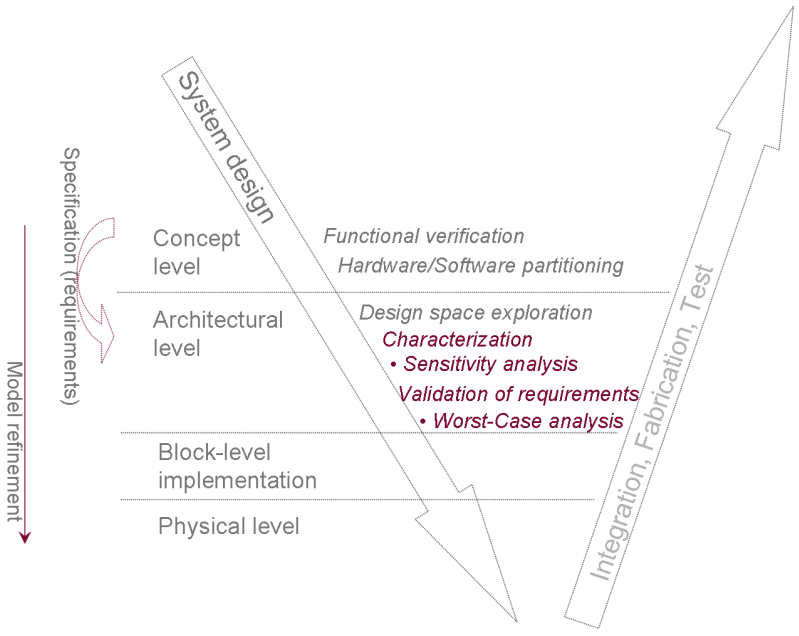


Figure 1.2: Phases of the V-model in ECU design and verification

A virtual executable representation i.e. a model of the system is built and refined as part of the design flow. Starting already from the conceptual level, it is often required to simulate it

[1] contain analog and digital electronics and embed software

in complete application scenarios or even with the surrounding environment and systems, e.g. for demonstration of customer applications. Model-based verification is deployed throughout the entire development process, but different levels of abstraction focus it on different requirements. Options for system-level modelling and simulation are detailed in the next section. The architectural level is of interest, when the system is already partitioned and modelled, and where the conformance of the virtual system against the requirements must be tested.

It is necessary to confirm that, given the current proposal of implementation, the system would behave as desired. The concept, translated into an executable description, must fulfill the target functionality for different application scenarios. This introduces a set of requirements coming from the target specification, e.g. expected ranges for system outputs, typical and extreme conditions in which the device must operate normally. Additionally, for the top level verification, nominal values and admitted variations of block-level parameters are provided as outputs of the concept level design and are also needed as inputs for the next step of block-level implementation.

At this level of system validation, sensitivity analysis is a step necessary for the design space exploration (Figure 1.2). It characterizes the impact the existing sources of variation have on the target behaviour, which is quantified in outputs of interest. The next step which is important for validation against requirements is the worst-case analysis. By worst-case we understand the situation farthest possible from the intended behaviour, with respect to the variable conditions.

Industrial verification rarely addresses these issues so early in the design process, i.e. at the architectural level. This is a gap which must be filled in order to avoid the situation where a system is functionally compliant, but fails the final tests of conformance, i.e. under specific settings of its parameters/stimuli, the outputs fall outside the required ranges. Manufacturers along the complete supply chain for automotive systems must commit to a high degree of dependability. Therefore, new methods are needed, which start from the basic simulation flow, and extend it to perform fast and reliable multivariable sensitivity and worst-case analyses. The inputs to these validation steps are the system model at the proper level of architectural refinement, where all variables of the system and environment are controllable, and the requirements with respect to these variables and target system behaviour.

1.1.3 Model-based design and verification

Model usage can accomplish in-depth verification tasks when hardware is not available, thus offers a cost effective and systematic solution to deploy key phases of the development flow. Some examples are: functional verification, architectural exploration, application demonstration, virtual platform software development. High-level models are needed in early phases, for concept definition and proof, but also along the way to tape-out, by reflecting the functionality of the complete system, in its real context of application. These conditions would be otherwise, i.e. in real hardware testing, hard to reproduce and almost impossible to explore.

Options for modelling and simulation

Figure 1.3, adapted from [54], gives an overview of current approaches for modelling digital, as well as heterogeneous systems, i.e. mixed-signal, mixed-level, mixed-domain. Specification and simulation of complete heterogeneous systems can be covered at a pure functional level by tools like MATLAB/Simulink. However, such tools do not cover architectural level details. On the

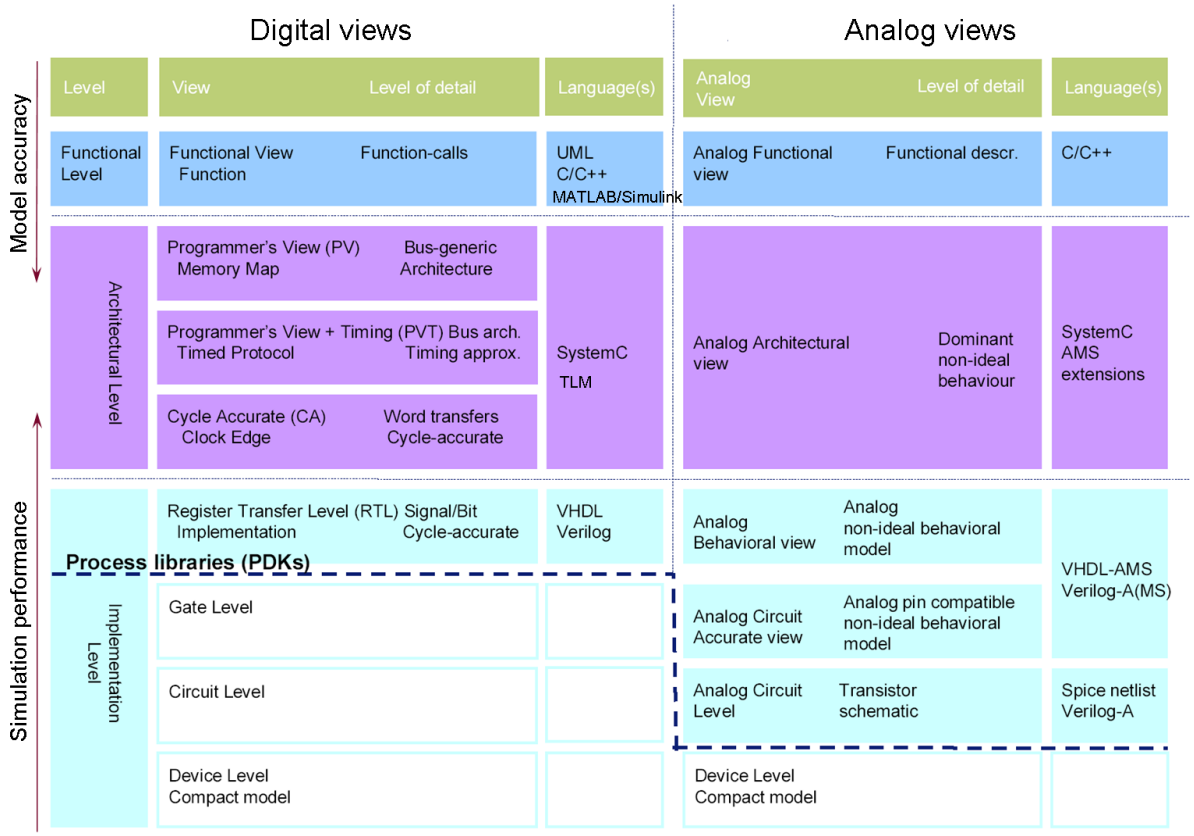


Figure 1.3: Specification languages and design abstractions

other hand, implementation-focused solutions like VHDL-AMS [PAT02] do not address the high complexity issues which occur in system-level modelling.

The above mentioned reasons motivate the use of SystemC [GLMS02], enhanced by its transaction level modelling (TLM) library [Ghe05], as description means which enables abstraction, and extended by SystemC-AMS [GBVE08], offering the possibility to describe heterogeneous systems. The extension has been continually improved [VGE05, VGE04], and proven its applicability in system-level modelling and verification [VPB+08, ADR+08]. As a consequence, first industry design flows are adopting it [63, 62], and provide reasons to approach it for the domain-specific problems we must face.

Adopting open source tools gives the opportunity to have interoperable models, and more perspectives to integrate them in a flow which suits best the needs of the respective field of application and level of abstraction desired. Additionally to the expressivity and flexibility SystemC can offer, a high performance in simulation can be achieved when the verification takes place at the right level of abstraction. This gain in speed is an essential advantage, which amplifies in the case of multi-simulation flows like the one proposed in this work.

System-level simulation for sensitivity and worst-case analysis

Simulation-based verification addresses a wide range of purposes, as Figure 1.2 also points out. The functional verification translates requirements on the behaviour into test cases and observes

how the system responds over the simulated time frame. In this case, it is of interest to check that specific events occur, e.g. power-up behaviour, switch of the outputs under predefined stimuli. Visual assessment is essential, while the specific timing or exact values of outputs are only roughly verified.

The next step must be to analyze these aspects deeper, i.e. apply changes on potentially variable aspects, quantify how close the result of the complete test is to the ideal one and interpret these values. This can involve optimization of the sources of variation, in order to get closer to the ideal behaviour, or modifying them to test the system by driving it as far as possible from the target behaviour. Either way, the output of each simulation becomes a very limited set of values as compared to a database of signals traced over the complete test duration. This reduced set of values symbolizes the system performance with respect to the simulated test case. It is referred to as the response, and exemplified in the next section.

1.2 Problem description

As previously mentioned, the main problem to deal with is handling numerous sources of variation with potential impact on the behaviour of the system. The values for the parameters of components, either internal or external to the system, or for the environment conditions, are not guaranteed, but rather vary in specified windows. Block parameters need tuning and their final values are still affected by manufacturing or operating changing conditions. Tolerances, safety margins or simply the room left for later design decisions always introduce variations. Hence the need for early validation of the system against requirements, in the presence of such influences [SB04b]. Responses and factors are illustrated in Figure 1.4 and are defined in the following.

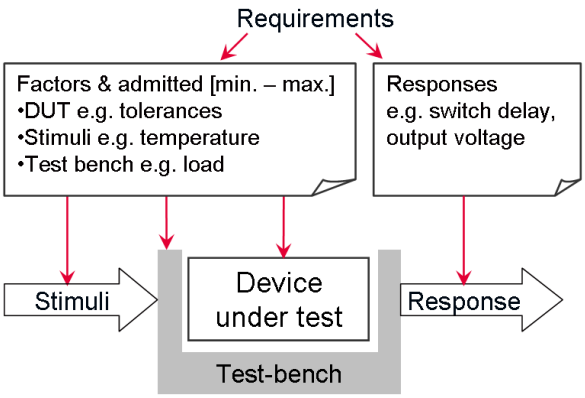


Figure 1.4: Factors and responses

Responses are defined as measures of the system's performance or characteristics at its interface, which must be within required ranges in order to validate the system behaviour. They are quantities which can be measured or computed after each simulation. Responses can be static, i.e. only one value characterizes one simulation run, with respect to what needs to be checked in the specific test. Common static responses are signal properties, electrical or timing-related e.g. maximum value, slew rate, drop in value within a predefined time. System response times, e.g. end-to-end transfer delays, time to process and transfer a message, can also be of interest and

be set as responses. A dynamic response refers to a set of samples for signals of interest, which quantifies its transient behavior. Dynamic response analysis is addressed later in the approach, as an extension to the basic static response flow.

Factors are sources of variation with potential impact on the response. As Figure 1.4 shows, factors can come from the stimuli e.g. the supply voltage, or can be parameters, both of external blocks e.g. loads from the test bench, and of the device under test e.g. component delays. Table 1.1 exemplifies factors with impact on common responses in ECUs, which have the typical structure introduced in Figure 1.1 [PGZ08].

Table 1.1: Typical ECU factors

Source of variation	*Factors*
Environment	Temperature, Supply voltage
Measurement	ADC resolution, Amplifier Gain, Offsets
Power switch	Slew Rate, ON-Resistance
Supply, voltage regulators	Reference voltage, Regulated voltage, Settle time
Supervision, Protection	Thresholds (temperature, voltage)
Transceiver	Jitter, End-to-end delays
Actuators	Loads

Factors are controllable inputs of the simulation process. They vary in pre-specified windows and form a multi-dimensional, continuous verification space. Experiments are sets of simulation runs which apply variations in the factors to identify reasons for changes in the response. It is necessary to optimally cover this hyperspace, to discover important factor effects and safety margins for system responses, to ensure the system conformance before moving on to the next design step. The characterization phase must realize a sensitivity analysis to determine how factors and their interactions impact the response. To validate that the response is within a specified range, the response's extremes over the verification space are of interest and they are referred to as worst-cases. Worst-case analysis explores the factor hyperspace to find the worst response and the main causes for it.

To reduce the overall verification time, great effort has been invested into developing modelling methods and solutions which speed-up simulation, as exemplified in the previous section. Raising the abstraction level gains performance in simulation, at the expense of a reduced model accuracy. But this is just a first step toward smaller verification times. Models can and should be resimulated until the conformance of the system is verified under the variation of all factors. This raises the question of which are good measures for the coverage of the factor space and for the reliability of the response validation in general. After defining them, the present work must find how to plan the set with minimum number of simulation runs, which is able to reach the target coverage and reliability in response characterization.

1.3 Objective and scope of the work

The general goal of this work is to perform reliable characterization and validation, at the architectural level as illustrated in Figure 1.2. In the context of this work, reliability is the probability of system conformance with value-ranged requirements. The performance of system validation is then given by how well the reliability has been evaluated. When many variables i.e. factors can

impact it and vary in continuous ranges, the ones with significant impact must be isolated and the worst-case behaviour must be estimated, to check that the system can be counted on under any condition. Therefore, the accuracy of estimates must be maximized for:

- the impact of factors on the response
- the prediction of the response in specific points, i.e. for specific factor settings
- the worst-case response

Since neither real effects nor the true worst-case is known, the error of response prediction in simulated points and the coverage of the factor space are to be evaluated as measures of performance. When statistical distributions of factors are known, biased sampling of the verification space achieves a higher, i.e. closer to real, coverage. Similarly, when the statistical distribution of the response is derived, any prediction made on its (extreme) values is improved.

In this process, it is important to keep the number of simulation runs under control. The number of runs is a difficult topic to address right from start, because most of the times a decision-based flow is needed, where simulations are run to improve the estimates of the previous steps, until the target reliability is reached. There is no hard restriction, but in general, no more than $1k$ runs should be involved, even for large factor sets i.e. of up to 30 factors. Therefore, an exponential increase with the number of factors is not acceptable.

The verification is simulation-based and the level of abstraction in modelling is rather high, i.e. the system is modelled using SystemC and its extensions. In order to address similar issues at a circuit-level, approaches are needed which involve a high computational complexity, e.g. to handle low-level thermal effects. These approaches are not applicable here because they cannot handle sufficient factors or they use too many simulation runs. This is justified more in Chapter 2 of related approaches and in Section 5.1 of implemented alternatives. Such complex approaches are not even necessary, therefore methods which take advantage of the higher level of abstraction are proposed.

Another aspect which reduces the scope of the approach is that the systems under study are designed for applications in the automotive domain. Most of the principles would be applicable to systems from other domains. The main aspects to consider are complexity reflected in a high number of factors i.e. more than 10 and heterogeneity of the systems: factors can be discrete or continuous, can have statistical properties and can be correlated to each other.

In addition to these boundaries for the scope of the work, which help to filter the possible approaches, the proposed approach presents limitations which reduce its area of applicability. These are detailed in Section 6.3.

1.4 Contributions and structure of the dissertation

The Design of Experiments (DoE) methodology is the starting point of the approach. DoE is a discipline with very broad application across many sciences. Classical DoE was initially focused on improving the outputs of real life experiments, by applying mathematical statistics on physical systems. For such systems, the scientific DoE started with agricultural experiments in the 1920s, followed by chemical experiments in the 1950s, and is now also applied in social systems [Kle08]. This domain is covered extensively by textbooks such [Mon05].

Out of the DoE classical framework, several principles apply only in real life experiments, while others work in simulation experiments as well. This class of concepts and methods has not been clearly delimited, especially because it is highly problem and domain dependent. The goals of experimenting vary in a wide range and the degree of complexity of the systems under verification also constrains the applicability of classical DoE. The degree of control exercised on the factors further differentiate the DoE applied on deterministic simulation models from experiments on random (stochastic) simulation models.

Therefore, what DoE covers is a wide variety of experimental designs and analysis methods, with a strong mathematical foundation and much past evidence that they can meet various design objectives. In addition, the so-called metamodelling framework provides an extensive class of deterministic functions (metamodels) which approximate the response with respect to factor variations. Results of properly designed experiments can accurately fit metamodels with power to predict the response in not yet simulated points.

What DoE and metamodelling domains do not cover, however, is to indicate how to isolate techniques which are efficient for specific cases. The gap between these domains, currently covered only in research, and simulation practices in industrial applications must be filled. The methods under evaluation can be proven effective if and only if they work on the complete, real-sized systems. Testing the methods on responses of deterministic systems or functions, i.e. where the worst-case is known or can be computed a priori, would not prove the applicability on our simulated systems. The systems must first be characterized in terms of response complexity and predictability, before establishing the equivalence to others. This calls for an empirical study, to reach a worst-case as close as possible to the real, unknown one. On the other hand, such systems are highly complex in terms of functionality, size and/or structure, thus they present large sets of factors and require long simulation times, so not many simulations can be spent.

The experiment flow proposed here manages to:

- find worst-cases and factor effects more efficiently than existing simulation-based approaches
- extend DoE to a strategy of sequential simulation, in order to build upper and lower bounds for the response, be it static or transient i.e. signals

which is a first in the high-level verification of automotive industrial systems and applications. More questions are addressed in the presented work: how to select the responses and the corresponding factors to test; how to implement the existing methods, since rather complex algorithms for statistical data analysis and optimization routines are needed; how to best integrate them with the system simulator; how to automate the flow, since many recommended techniques are decision-based, so they involve the user interaction; whether and how they can be coupled with other deterministic optimization methods.

The present work extends the basic simulation framework to implement the core of DoE. Classical methods for experiment planning and analysis are first tested and compared on simulated systems, then optimized, to be efficient in this extended framework. Predictive metamodels are built and optimized to characterize the response and replace expensive simulations. Worst-case estimates and factor effects are derived from the metamodels, corrected and confirmed by simulation. These are improved by the extension referred to as random factor experimental design, in order to account for factor distributions and conclude on the resulting statistical distribution of the response. A reduced number of runs is invested to extract the necessary results, in terms of coverage and accuracy of the estimates for factor effects and for the worst-case.

Chapter 2 realizes a selection out of the existing problem-related approaches. It explains the basic concepts and justifies the choices of DoE and metamodelling as starting point. Chapter 3 implements them and evaluates their feasibility. Section 3.1 presents a solution to implement multi-simulation flows. It uses standard tools, to build a reusable, transparent framework, suitable for experiment planning and analysis. Section 3.2 details selected principles of experimental design, while Section 3.3 adapts methods of statistical analysis of the experimental results. Section 3.4 shows how to interpret the final outputs of the analysis methods, in order to reach the targets initially defined.

The complete experiment is automated in Section 4.1, so that it can become a sequential, self-adaptive flow, with ability to revise assumptions evaluated false and to reuse the previous simulations' results. Section 4.2 extends the basic experiment flow to iteratively simulate in the region of the worst found case and increase the reliability in the worst-case estimate.

The automatic experiment flow is robust enough to be extended for a transient analysis, where response metamodels are estimated for several sample times along complete simulated test cases (Section 4.3). This is of interest because the impact of factors and the worst-case drivers change in time. The sensitivity and worst-case studies span over complete simulated time frames, and do not even require more runs than before.

The methods require minimum additional effort in preparing the model of the system and in building the experimental framework. The applicability of methods, in terms of conceptual reuse and portability from one case study to another, is maximized. A configurable interface is offered, which requires no more interaction than needed to the user, and provides sufficient outcome of the experiment. The approach is concluded efficient only after evaluating a high overall performance, on the results of two case-studies: a window lifter ECU and an airbag control system. Several alternatives are implemented for comparison.

Table 1.2 represents the structure of the proposed approach, along with its extensions. The first two columns point to chapters/sections which present methods or results, while the issues which are solved in the respective part are indicated in the third column.

Table 1.2: Structure of the approach, and addressed issues

Chapter	*Section*	*Solved issues*
Adapted DoE flow	Preparing the experiment 3.1	• configure and control multi-run experiments 3.1.1 • extract from the requirements the [factors, response] sets 3.1.2 • setup the model to run the experiment 3.1.3
	Designing the experiment 3.2	• evaluate the feasibility of the DoE principles in simulation experiments on the systems under verification 3.2.1 • select the metamodels which can fit the responses 3.2.2 • identify and adapt classical experiments which can meet the requirements e.g. as number of runs 3.2.4 • account for statistical properties and correlations of factors 3.2.3, 3.2.5
	Building the metamodel 3.3	• adapt statistical DoE methods to postprocess the results • fit the metamodels for accurate estimates 3.3.2
	Concluding the experiment 3.4	• extract and interpret factor effects 3.4.1 • characterize the response in terms of statistical properties and the worst-case 3.4.4.2, 3.4.3
Extensions of the DoE flow	Sequential experiment flow 4.1	• extend the flow to an automated sequence • revise assumptions on the response • reuse results from one experimental step to the next
	Optimization of the worst-case search 4.2	• gradient-based search to optimize the worst-case estimate
	Transient response analysis 4.3	• extend the flow to analyze more time samples • reduce the overall postprocessing time
Results	Alternatives to the DoE flow 5.1	• implement existing approaches to address similar issues • characterize their performance for comparison
	Window lifter system 5.2	
	Airbag driver system 5.3	

2 Related work

This chapter discusses existing approaches to similar problems, both classical and recent. Concepts, terminology and methods related presented approach are also introduced. The first section classifies the related approaches. Then, the main classes are separately treated, while the last section explains the the existing gap between industrial practices and research, which must be covered.

2.1 Classification

The problem is defined in relation to the scientific terminology commonly used. The general goal is to find a set of controllable inputs (factors) which minimizes a given objective function (response). Since the response is not available directly and must be estimated through simulation [FGA05], it is a simulation optimization problem. In a stochastic problem, the simulation provides a noisy estimate for the response, and replications of simulations yield better estimates for the response expectation [FGA05, Kim06]. The present problem is a deterministic one, however, as the response is uniquely determined given a fixed set of parameters. Although a continuous optimization problem, because of continuous factors' ranges, it can be considered discrete, with levels for the factors obtained after discretizing these specified ranges.

Figure 2.1 shows a rather simplified hierarchical description of the present approaches to similar problems. [FGA05, FCS08, SPKA01, WS07, MM06] contain reviews on such approaches, often also referred to as simulation optimization techniques.

The main differentiating criteria is whether the strategy is sequential (or adaptive) i.e. whether outputs of previous simulation runs are fed back to the simulation controller. This splits the approaches into directed test, i.e. where no feedback exists, and adaptive search. The third option of symbolic simulation, when feasible, would not need a feedback loop because it covers the input space within one simulation, by symbolically representing variations of factors. Details are provided in Section 2.6.

The second criteria is whether statistical properties of factors are used to generate the values which are tested in simulation: random test (or Monte-Carlo methods) consider these properties. Such methods, whether adaptive or not, are analyzed in Section 2.4. A selection of these is adopted to be extended by the presented work.

The genetic algorithm approach and the rest of meta-heuristics are also covered separately, in Section 2.5. The part which presents most interest is the metamodel-based class of methods.

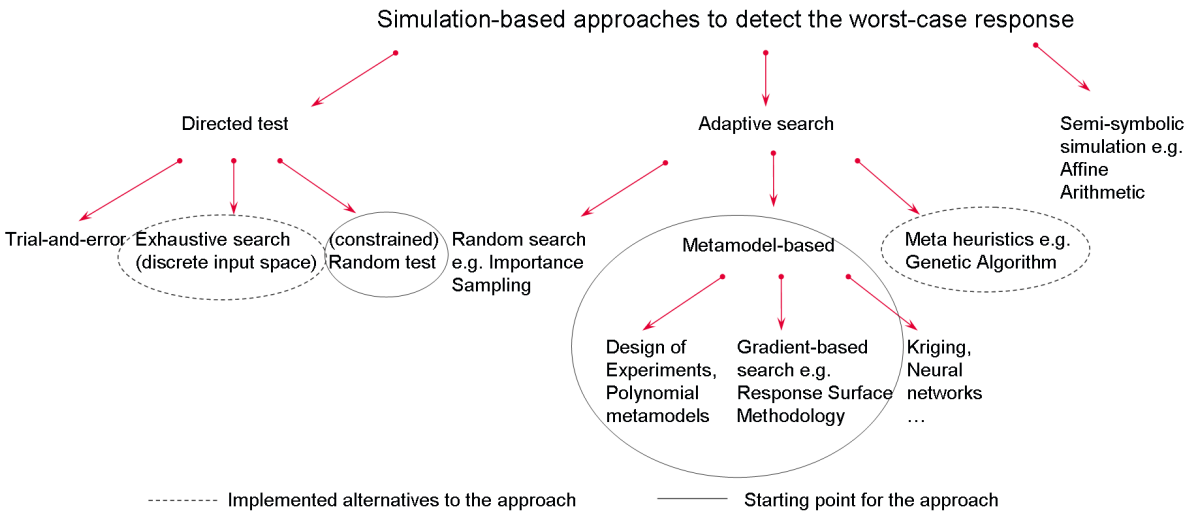


Figure 2.1: Related approaches

This consists of the core, i.e. Design of Experiments methodology, treated in Section 2.3.2. The Response surface methodology, as a representative of the gradient-based approaches, is also a metamodel-based method. It is discussed in Section 2.3.3.

2.2 Directed test verification methods

Directed test methods apply predefined settings to factors. This class includes the nominal value simulation, when each factor is set to the typical (or central) value of its range. This is the starting point of any simulation flow, to ensure the system behaves as expected under nominal simulation settings. Unfortunately, these methods are often the only ones used in practice. But for in-depth analyses of the system's sensitivity and worst-case, directed test is less efficient, as compared to approaches which learn about system behaviour from the previous runs. The worst-case is the worst simulated case, because of no predictive nature of the process.

Trial-and-error methods such as best guesses fall within this class of directed test. They are not reliable, since they assume knowledge of the system and are subjective. One-at-a-time approach successively varies each factor to observe the impact on the response. The problem with this method is that factors can interact in their impact on the response, thus must be varied concomitantly to draw correct conclusions.

Exhaustive search methods simply test all factor combinations and are applicable only when factors have a finite number of levels. A proper discretization (or sampling) of the factor space and exhaustive exploration of the sampled space (when feasible) ensures a higher quality for the worst found response than trial-and-error methods. This is rarely used, because of the high dimension of the factor space. For instance, uniform grids present an exponential increase of the runs with the number of factors n and a power increase with the number of levels per factor k: $number\_runs = k^n$. Even for a poor discretization of $k = 2$, i.e. each sample is a corner of the factor space, the method requires more than $1k$ runs for $n = 10$, and more than $1M$ runs for $n = 20$.

Random or selected corners can reach a good performance only when the control algorithm is sequential (adaptive), as implemented in [SREP08]. Otherwise the methods also fall in the category of trial-and-error, and are considered inefficient. All of the above methods are simple to implement, but are low on performance. They either ensure unacceptable coverage or they require too many runs.

Space filling experimental designs target an optimal discretization of the continuous factor space, by a given number of samples. Even when used as directed test methods, they can achieve a better coverage then the ones previously mentioned. However, such designs should be involved in metamodel-based approaches, i.e. be succeeded by a step of metamodel fitting. They are covered in Section 3.2.5. Random test methods, although trial-and-error, are separately treated in Section 2.4.1, and are adopted by the approach.

2.3 Metamodel-based methods

Metamodels represent deterministic approximations of the response of the system with respect to the factors under variation, over the space of interest. They are also referred to surrogates by sources such as [GTCD09]. Once the functional behaviour has been validated[1], metamodelling can serve one or more of the following objectives [Bar04, Mon05]:

- Screening: in the early experimentation phases, the number of the factors can be reduced, by identifying and removing the ones with insignificant effects.
- Sensitivity analysis (or characterization): by choosing an intuitive representation of the metamodel with respect to the influence of factors and their interactions, the metamodel can assist in understanding the impact of factor variations on the response.
- Design space exploration (or model approximation): based on a set of underlying assumptions, an interpolation is performed to predict the response between the simulated points. Evaluating this fitted metamodel, i.e. the interpolation, in specific points is then an inexpensive alternative to simulation.
- Worst-case analysis: the metamodel is used to predict where the response reaches extreme values, i.e. is farthest from expected.
- Design optimization and robust design: for design optimization, changes are applied in factors to identify the settings which best accomplish the functionality. For robust design, the target is to choose factors which ensure minimum variability in the response.

In the context of the present work, the experiments must characterize and validate the system, according to the requirements described in Section 1.1.2. Screening filters the initial set of factors to a reduced subset, which can be efficiently handled by experiments with a reasonable number of runs. Sensitivity analysis determines the impact of the remaining factors on the response, by fitting the simulation results on a metamodel equation. Then, the worst-case response is searched with the help of the fitted metamodel, in order to ensure conformance to the requirements or, alternatively, to explain potential nonconformance sources.

[1] The nominal setting of the factors ensures a system behaviour as expected

Although metamodelling is performed, factors are not varied to find optimal implementations, i.e. for system optimization (usually done over wide factor ranges). Nor are factor values searched, to ensure less response variability, i.e. for robust design.

2.3.1 Metamodels and fitting methods

Depending on the dependency on the factors, metamodels can be multivariate polynomials or have more complex forms. When the specific representation is parametric, it needs estimation of the parameters' values. For this, an experimental design (or sampling algorithm) is built, which requires minimum samples of the factor space, and provides optimal data to estimate the metamodel parameters, i.e. to fit the metamodel. The sample size increases with the number of parameters to estimate, therefore with the metamodel complexity. The main challenge to finding an appropriate sampling algorithm is, however, the increase of the sample size with the number of factors. To use the metamodel to characterize the dependency response(factors) and to predict the response in points of interest, a step of metamodel validation must be performed. This evaluates how close the estimates provided by the metamodel are to the response values extracted from the real system. Various metrics on evaluating the residuals, i.e. distances between predictions and response values, with associated thresholds, are available: [GCLD10]. Optionally, a step which optimizes the parameter estimates can follow.

A wide variety of metamodels, with associated sampling strategies, as well as fitting and validation methods have been covered in research [SPKA01, WS07]. For the present approach, polynomials are chosen: they are simple parameterized metamodels, which realize an intuitive characterization of the response by attributing at least one coefficient to each factor and to each interaction of factors. This is found sufficient to approximate the response on the regions under study with acceptable residuals. Such metamodels can handle enough factors and are flexible enough to allow corrections of estimates and reuse of experimental results from one step to the next.

Metamodelling strategies which perform a more accurate interpolation of the simulation results are available. For instance, the Kriging metamodel, popular in deterministic simulation experiments, corrects the polynomial approximation by an additive term Z. Z is assumed to be a realization of a stochastic process in which the covariance structure relates to the smoothness of the response. Several correlation functions between sample points can be chosen. Software packages like [56] estimate the Kriging parameters. However, fitting such a metamodel involves more estimation overhead (complex regression algorithms), thus more potential sources of errors. Although these metamodels can achieve higher accuracy, perfect interpolation of the simulated points is not considered necessary.

Even more sophisticated metamodels need more effort to be built, as number of simulations and algorithmic complexity. Less intuitive response representations are used, therefore factor effects are harder to be extracted and interpreted. They involve more parameters, therefore more necessary runs. For instance, the sampling which is recommended in [KR03] starts with a grid of 3 levels per factor and assumes a number of factors smaller than 6. [56] also suggests grids as space filling designs. But, as mentioned in Section 2.2, this exponential increase of runs with the number of factors is not acceptable.

2.3.2 Design of Experiments

DoE is an approach to plan and analyze real life as well as simulated experiments [Mon05]. DoE

is a metamodel-based strategy, with particularities in each of the steps previously detailed i.e. experimental design, metamodel choice, metamodel fitting and validation. It has emerged in many fields, as it invests a reasonable number of simulation runs to handle a highly dimensional verification space, while still locating factor effects unlike other methods [Bar04]. DoE is applied and extended in the present work for deterministic simulation experiments, where all factors are controllable and under investigation, to study the impact which functional blocks and their interdependencies have on the system outputs. This leads to an efficient, still reliable, multi-run strategy to discover response bounds and understand the main causes for worst-case behaviour.

DoE concepts

DoE assumes that the response dependency on the factor set f can be approximated with a multivariate low-order polynomial metamodel $R(f)$:

$$R(f) = c_0 + \sum_{o=1}^{2} \sum_{i=1}^{n} c_i^{(o)} \cdot f_i^o + \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} c_{jk} \cdot f_j \cdot f_k$$

Its coefficients quantify the effects of factors on the response: $c_i^{(1)}$ - main effects, $c_i^{(2)}$ - quadratic effects, c_{ik} - 2-factor interaction effects. The assumption is based on the sparsity of effects principle which often applies in practice, that implies the system is likely to be driven primarily by some main effects and low-order interactions.

Statistical DoE plans experiments with minimum runs, in order to find factor effects and how they interact, with maximum statistical confidence. The term DoE is used to refer to such experimental designs. Multiple regression extracts the coefficients which best fit simulation data. Orthogonal DoEs allow optimal regression of results, because they obtain results which enable a decoupling of factor effects. These DoEs work in the assumption that estimated effects are significantly higher then the effects which are not taken into account. This is reasonable according to the sparsity of effects principle, and can be checked after regression.

2-level factorial DoEs set all factors to either the minimum or maximum of their ranges. Such factor sets are referred to as corners. 2-level fractional factorial designs consist of selected factors' corners, and are widely used to investigate main and interaction effects. Response Surface DoEs additionally estimate 2^{nd} order effects, using at least one extra level per factor. Space filling DoEs can improve the metamodel fitness at the cost of more simulation runs. E.g. a Latin Hypercube Sampling (LHS) generates random factor levels, e.g. normally distributed, in order to maximize the minimum distance between points in the factor space [56]. Software packages generate the tests required by such DoEs [52, 53]. To evaluate the fitness of the regression model as a response estimate, the set of residuals is analyzed. For adequate models, they must be small enough, approximately normally distributed with mean zero and not correlated to the response values.

A classical experiment flow includes the following steps [Mon05]:

1. Define the objective of the experiment
2. Define the response
3. Choose the factors and their levels
4. Plan the experiment
5. Execute the experiment

6. Analyze the results
7. Conclude

More details are provided throughout Chapter 3, where a selection of the available methods is made to support the approach. Section 3.2.1 presents how principles of experimentation are adapted to the particularities of the problem, while Sections 3.2.4 and 3.2.5 adopt a set of the classical DoE methods to use in the approach. The results of these DoEs are analyzed using classical and extended methods, as well as custom ones, introduced in Sections 3.3 and 3.4.

Applied DoE

Several recent applications support DoE methods for input space exploration and estimation of effects. [MPLM07] exemplifies application of DoE in practical simulated experiments, but states some problems. E.g. the experimental framework should be designed for simulated systems, instead of adapted from real world experimentation cases, as they differ significantly. Moreover, there is a recognized gap between simulation practitioners and applied DoE theories.

DoE can control simulations of parameterized systems, for various verification purposes: screening [TM01]; sensitivity analysis [NYLS05, SB04a]; for robust design [ATW06]; for multi-objective (multi-response) optimization [TWLBX09]. Previous work evaluates experimental designs to optimally cover the verification space or reduce the number of simulation points while keeping reasonable prediction models [San07]. Simulated DoE was efficient in more areas of electronic system design: tuning microprocessors [SVL07]; designing chip floorplanning [NYLS05]; CMOS technologic processes [SB04a]. However, most applications are either limited to screening, or deal with relatively few factors (<10).

The present work is supported by publications of the author in field-related conferences: [PR10, RDGP10c, RDGP09b, RDGP10a, RDGP10b, RGDP10, RDG+10]. They show how experiments were conducted on automotive ECUs, modelled using SystemC and its extensions. But more on the topic is presented starting with the next chapter.

2.3.3 Response Surface Methodology

The Response Surface Methodology[2] is a gradient-based technique which addresses deterministic, continuous optimization problems. The procedure identifies the direction of the maximum increase of the response. A local response surface representation is built, which drives the sequential search for the response extreme. Given a current best setting of the input variables, a movement is made in the gradient direction. The two common representations are regression models and neural networks. Optimization is then applied using deterministic optimization procedures. Linear regression can guide estimating the direction of the steepest descent. Once the region of optimum has been narrowed down, a more elaborate model is estimated to lead the search [Mon05, FGA05].

The main drawback is that the regression assumptions must hold. It presupposes that the response variable is differentiable in the feasible region, with respect to the inputs. In addition, it is best suited to find local extremes, and it is difficult to apply on a highly dimensional input space [FGA05]. One reason is the large amount of simulation points in one area before exploring

[2] also known as "hill climbing" or method of the steepest ascent

other parts of the search space, which increases with the number of factors. Moreover, commercial software for simulation optimization does not integrate the simulation model and the optimization routine [FCS08, FGA05].

Gradient-based methods are coupled to the experimental framework and applied later on. The algorithms do not scale when applied on the initial factor space, as observed in the results. A low convergence rate is caused by the assumptions of regression algorithms i.e. continuity, derivability, which do not hold over the full initial space. Even more, it isolates the local optima instead of targeting the worst-case, i.e. global response extreme. However, it can conduct an efficient search in a relatively small region, when it is applied on a reduced area in the neighborhood of the previously estimated response extreme. The implementation is detailed in Section 4.2, while Sections 5.3.5 and 6.1 provide results of such an implementation.

2.4 Random test methods

Basic random test is commonly referred to as Monte-Carlo simulation and is a type of directed test. Random search, on the other hand, assume using the results of the previous simulations, to guide the worst-case search. Both of them are detailed next.

2.4.1 Monte-Carlo methods

The classical method is a circuit-level analysis which randomizes prior to the simulation technology parameters of the device models reflecting fabrication variations, e.g. doping concentration, and observes their effects. Monte-Carlo methods can in this sense be extended to the system level, in order to determine effects of statistical variations of factors on the simulation responses. The controlled randomization can account for dependencies between inputs e.g. by deterministic correlations or other constraints. Monte-Carlo and constraint-random verification are applied in practice, e.g. in [NZH+08]. Most of the times, these methods are limited to low level verification, e.g. circuit or block levels.

For system level randomization, the SystemC Verification Library [55] offers efficient implementation of constraint-based random stimulus generation [GED07] and it brings interesting ideas about how to build custom randomization functions. An extensive set of classes is offered, and the coupling to the simulator should present no problem. However, it does not support directly generic distribution functions, which are common with other packages like MATLAB, and which are considered necessary for the approach. Moreover, it does not provide a built-in simulation multi-run ability. The formulation of correlations between factors is found of use for the present goals and is analyzed in more detail by the approach.

The proposed methods use as starting point random test methods, implemented using the statistical MATLAB functions [52]. The statistical package presented in [MHE08], which implements system-level Monte-Carlo simulations on SystemC models, is also used. These are extended to handle factor correlations, and postprocessing steps are added to address sensitivity and worst-case analysis. This type of experimental design is referred to as random factor DoE (RFDoE).

2.4.2 Random search and Importance Sampling

Random search algorithms address problems of discrete input space exploration. They move iteratively through the feasible parameter space, like gradient-based procedures, but instead of using a gradient, the next candidate solution is probabilistically drawn from the neighborhood of the current best. The success depends heavily on the defined neighborhood structure [FGA05].

Methods like Importance Sampling are somehow related to random search. Their underlying idea is that certain values of the random variables in a simulation have more impact on the response than others. If these important values are emphasized by sampling more frequently, then the estimator variance can be reduced. It is important to choose a distribution which encourages the important values. This use of biased distributions results in a biased estimator for extreme response values [59]. [SR07] synthesizes such ideas from Importance Sampling, data mining and Extreme Value Theory. It demonstrates the application of Statistical Blockade, to overcome simplistic assumptions about worst-case corners, and speed-up standard Monte-Carlo.

Such approaches involve too many simulations ($> 1k$) to deal with the number of factors required in the systems under study. These methods usually address circuit-level problems, while for system-level simulations assumptions on the response e.g. only low-order factor effects, need to be made and checked, to simplify the problem and reduce the number of runs. In addition, the approaches are strictly worst-case oriented and do not make use the simulation data to realize a system characterization. The sensitivity analysis problem should also be addressed while exploring the verification space. However, [SR07] offers interesting ideas, e.g. how to build and apply filters to the input variables before simulation.

2.5 Evolutionary algorithms

Worst-case search is also addressed by deterministic metaheuristics. This is a category including approaches such as genetic algorithms, tabu search, scatter search and other iterative and population-based algorithms. Extensive reviews are offered by sources e.g. [FGA05, FCS08].

Concepts of genetic algorithms

The genetic algorithm (GA) is a method for solving optimization problems by repeatedly modifying a population of individual solutions. At each step, GA selects individuals randomly from the current population to be parents and uses them to produce the children for the next generation. GA is commonly applied to solve problems that are not well suited for standard optimization algorithms. To create the next generation from the current population, GA uses

- selection rules to select the individuals that contribute to the population at the next generation, i.e. parents
- crossover rules, to combine parents to form children for the next generation
- mutation rules, to apply random changes to individual parents to form children

Unlike gradient-based algorithms, GA generates a population of sample points at each iteration, and the best point in the population approaches the solution. Another important difference is that GA involves random generation when selecting the next population, i.e. not only deterministic computation [51].

Applied genetic algorithms

[SREP08] presents a study of worst-case response time estimation of distributed real-time systems, for automotive communication protocols, simulated with SystemC. The optimization strategies in [SREP08] involve GA and bring out interesting ideas of corner-case reduction. The addressed problems are typical to real-time systems i.e. message scheduling policies for processes in communication protocols. The present goals concern, however, heterogeneous systems with multi-nature factors and system inter-dependencies, which must be abstracted to reduce the number of runs.

Combining evolutionary algorithms with the Response Surface Methodology has also been under research. A drawback is the assumption that the input space is of low dimension [Guo07]. In general, such algorithms are necessary for highly complex responses. [KO07] also presents an approach applying some of the mentioned techniques i.e. meta-heuristics, random search. As viewed by [Guo07] as well, these reasons make evolutionary algorithms not applicable when many factors must be handled. Moreover, the sensitivity analysis problem should also be addressed. The present work tests GA and compares its results to a gradient-based worst-case search, and finds a lower performance for the GA search.

2.6 Semi-symbolic verification

The semi-symbolic simulation is the counterpart of formal verification in the analog-mixed signal domain. The Affine Arithmetic approach[3] is a representative, which optimizes classical interval arithmetic. It can be successfully applied on systems with uncertainties, in order to estimate worst-case values of signals over the simulated time frames [GHW04]. Since an exhaustive search of the continuous factor space is not possible, Affine Arithmetic symbolically represents variables in the simulation process. Because symbolic simulation can cover many system executions in a single run, it can greatly reduce the size of such verification problems [61].

Methods based on Affine Arithmetic lead to safe, but over-pessimistic response bounds when dealing with complex effects and interactions, and are hard to apply on systems which need to transfer discretized information between different blocks. In addition, documentation and software implementations are hardly available.

[FS00] applies this approach to compute outer bounds and GA for inner bounds of the response. Although efficient for the addressed topic, it would present limitations in the present case, with respect to the number of uncertainties and width of the ranges to cover. As the authors conclude:

> "much care is required as the computation time could become unsustainable, as is dictated by the joint action of several factors: the number of uncertain parameters, the width of relevant uncertainties, the use of iterative models, the length of computation chains,the number of nonaffine operations involved by the evaluation function. For this reason, trying to define precise limits, in terms of maximum number of components or degree of complexity or any other factor, for the circuits that can be analyzed by means of AA is meaningless." [FS00].

[3] Affine arithmetic is a model for numerical analysis, which represents the quantities of interest as affine combinations (affine forms) of certain primitive variables. These variables stand for sources of uncertainty in the data or approximations made during the computation.

The application domain is in this sense restricted to the circuit level:

> "A tolerance analysis method is presented, which is best suited for studying circuits represented by iterative models/equations or whose response is known in explicit analytical iterative form." [FS00].

To address a higher level of abstraction, techniques which make use of the simpler underlying models of computation are found more time effective and sufficient in terms of worst-case results.

2.7 Open issues

Approaches which are common among simulation practitioners are often limited to directed test methods: nominal value simulations; best guesses, e.g. selected corners; one-at-a-time approach; exhaustive search e.g. by full corner-case exploration. Monte-Carlo and constraint-random verification are applied in practice, but most of the times only at a low level, e.g. circuit level verification. However, in none of these cases a proper analysis of simulation results is made and information useful for system characterization is wasted. As a consequence, pessimistic guesses on the effects of input variables are made in order to ensure the specification compliance in the worst-case. These lead to over-estimations, i.e. oversize the specification range, over-restrict the inputs' ranges [Law07].

For the reasons stated in this chapter, metamodel-based methods, starting from the core of DoE, indicate the most appropriate direction to follow. When it comes to such approaches, there is a significant gap between simulation practices and the research community. Implementation and test on real-sized applications are needed. From a practical point of view, the routines for experiment planning and analysis must be coupled to the simulator. Automation of the complete flow is then to be implemented. The flexibility and reusability of the algorithms must be maximized.

The availability of software to implement the DoE or metamodelling approaches on an industrial scale is arguable: MATLAB toolboxes provide support for generation of DoEs, but only for real-life experiments. Still, the functions from the Statistics toolbox [52], together with optimization algorithms [51] can be adapted. Software packages such as the Design and Analysis of Computer Experiments toolbox [56], the SUMO toolbox [GTCD09] (also MATLAB-based), or [53] also provide support for experimental design generation and metamodelling. The dimension of the factor space always represents a restriction. This is discussed more in Sections 3.2 and 6.3.

From a methodological point of view, the approach must first evaluate whether the classical framework provided by the statistical DoE is applicable to the presented problems. Then, experimental designs and metamodel representations which are effective must be identified. Algorithms which can fit and validate the metamodels must be adapted and extended when it is needed. In addition, the framework must be extended to account for factor distributions and correlations, because they are important aspects for the variations of real systems. Finally, the results must be analyzed and processed to compute the effects of factors and predict the worst-case, and properly presented to the verification engineer e.g. by special graphical representations.

3 Adapted DoE flow

This chapter details how the present work deals with the problems described in Section 1.2. The core of the approach consists of the main phases to deploy a simulated experiment, which are described in Figure 3.1. The chapters which detail their implementation are also referenced.

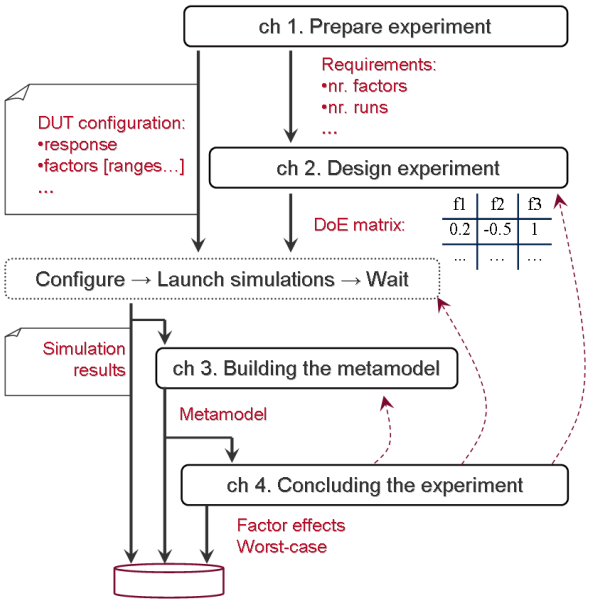


Figure 3.1: Main experimental flow

- Preparing the experiment, Chapter 3.1. The setup and control of the simulations is also described: Section 3.1.5.
- Designing the experiment, Chapter 3.2.
- Building the response metamodel, Chapter 3.3.
- Concluding the experiment, by characterization of the response with respect to impact of factors and the worst-case: Chapter 3.4.

Chapter 4 will present extensions of this flow, which represent a big advantage of implementing the previous steps.

3.1 Preparing the experiment

This chapter focuses on the prerequisites for the next phases of the experiment. These preliminary steps must convert the objective of the experiment into inputs which the algorithms can interpret. Outputs of the simulation must be filtered and properly transferred to the postprocessing algorithms. The analysis of results must finally translate the conclusions to the user.

Section 3.1.1 introduces the functional parts of the simulation experiment (Figure 3.2) and their interfaces, as well as the experimental flow (Figure 3.3). Section 3.1.2 shows how to transfer the objective of the experiment into executable descriptions. Section 3.1.3 provides information on the device under test (DUT) model, with its associated test bench and stimuli structure. A modelling approach is recommended, in order to easily configure and monitor the simulation model. Then, the simulation flow is detailed. Section 3.1.4 describes requirements on the first run, both from the functional verification and from the experiment implementation points of view. Section 3.1.5 describes the simulation controller and the communication scheme to the user and the simulator.

Therefore, this part contributes to the approach by ensuring that any specification requirements can be passed to the experiment planning algorithm. Starting from standard tools of simulation and data processing, a framework is built which automates the DUT setup and experiment simulation, together with collection of necessary data for analysis of the experiment. These phases of preparation of the experiment and simulation control can be decoupled from the experiment planning and analysis of results, which will go hand in hand and will be detailed in subsequent, dedicated chapters.

3.1.1 Experimental framework and basic flow

Figure 3.2 represents the basic elements of the setup for the control of multiple simulations required by the experiment.

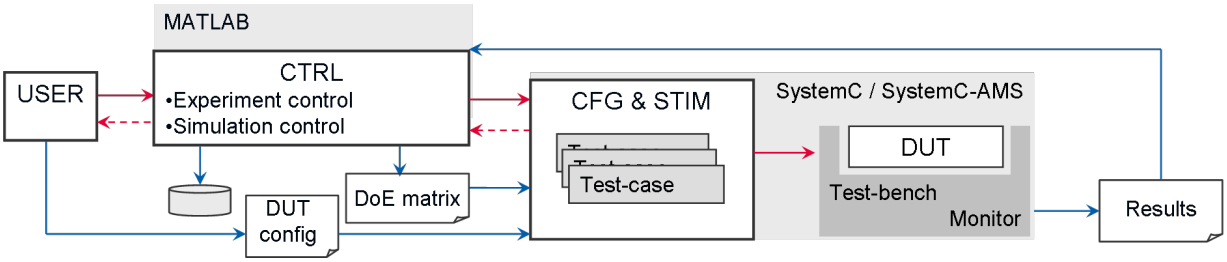


Figure 3.2: Experimental framework

The DUT refers to the model of the system under study. For each simulation, it must be set up in a test bench, configured and stimulated according to the experiment-specific test. A monitor unit must record the response values for the chosen experiment. It must also be configured prior to the simulation, in order to store no more than the necessary data.

The configuration and stimuli unit (Cfg&Stim) is responsible for these steps. It transfers the input from the controller (Ctrl) and from the DUT configuration file (DUT config) to the model. This file contains the current setup of the DUT and of the monitor. More about the configuration file can be found in Section 3.1.2. The results file consists of the response values recorded during the experiment.

The controller plans the experimental design, drives the simulation and analyzes the results. The DoE matrix is the output of the experiment planning step, performed by the controller. It is a $m \times n$ matrix, which specifies the setting for each of the n factors, in each of the m simulation runs of the current experiment. The communication to the controller can be differentiated into a "control flow" and a "data flow". The control flow (marked by red arrows in Figure 3.2) refers to getting the user input, as well as launching the simulations and receiving the simulation status from the simulator. The data flow, marked by blue arrows, is the transfer of configuration and DoE matrix data, as well as simulation results and experiment reports.

The controller reports to the user which point of the experiment has been successfully passed, to easily monitor the progress (response values which are successfully recorded, runs which are simulated/remaining/failed, partial results of the analysis step). Simulation bottlenecks or postprocessing steps which take longer than expected can be then detected to terminate the experiment, if necessary. The final report is also stored by the controller.

Some choices relevant to this framework are discussed in the following.

There must be a single source to provide the settings for the factors active in the current experiment, necessary for the DUT (re)configuration. These settings must however be visible to the user. A good choice is to apply them directly from the controller. On the other hand, the controller must be decoupled from application dependent details, such as nominal values of factors. Therefore, a configuration function, contained in the Cfg&Stim unit, is used to translate the data of the current test into DUT specific settings.

The controller unit will pass the reconfiguration data for each simulation run in the form of the DoE matrix. It acts as a master, controlling the simulation executable. Alternatively, the controller could forward the complete DoE matrix to the Cfg&Stim unit, with the advantage of less data communication between units, i.e. only once for a larger set of runs. That would imply, however, less control and insight from the controller side, and additional overhead to launch the complete experiment from the Cfg&Stim side.

Another important aspect is what and (from) where to monitor and report, with respect to the simulation results, so that the controller receives no more than the experiment specific information to be postprocessed. A monitor unit inside the test bench will be responsible for that. Details can be found in Section 3.1.3. Generally, the implementation targets an overall increase in performance, by minimum time and memory space consumption and minimum file handling. This lowers the risk of errors and the overhead to access configuration/results/report data.

Figure 3.3 represents the experiment flow, focused on the communicating entities.

- The User sets the objective of the experiment. This is transferred through the DUT configuration file, but also by passing options to the controller. Section 3.1.2 describes how requirements, extracted from the specification, are translated into experiment setup.
- The nominal simulation, i.e. with no variations in the factors, is first executed and checked, as detailed in Section 3.1.4. Then, the controller generates an experimental design, according to the configuration file and other options set by the User.
- (Loop for the number of runs) Each simulation run is launched, according to the DoE matrix file. The Ctrl executes the Cfg&Stim, by passing the test case of choice and the index of the current run. The Cfg&Stim unit configures the model using the settings from the DUT configuration file. During the simulation, the monitor stores the response in the results file, then stops the simulation if configured so. The stimuli function returns the status to

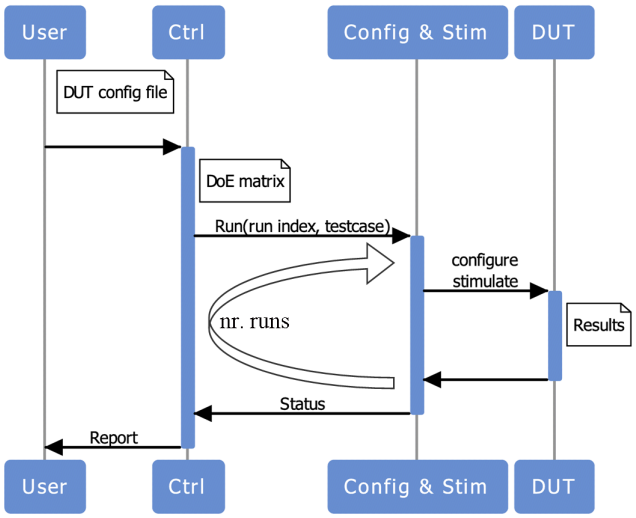


Figure 3.3: Simulation control flow

the Ctrl. More details about the model setup are provided in Section 3.1.3 and about the simulation control process in Section 3.1.5.

- The controller collects and postprocesses all simulation data. The final report is then provided to the user.

3.1.2 Response and factor definition

This section explains how the objective of the experiment must be transferred to a format which can be interpreted for experiment setup and processing of results. Requirements on the system behaviour under specific stimuli, and variabilities with potential impact on it are extracted from the specification. These must be converted into the response, respectively factors' definition. Factors are directly mapped to parameters of the units of the DUT model, and responses to signals in the test bench or related events.

Specification information must be extracted, filtered and centralized into the DUT configuration file. A prerequisite of the experiment is that the ranges for the factors are or, if not, they should be specified by the initial requirements, so that the user can simply extract the relevant information. Another prerequisite is the DUT model exists and that all the variables, potential factors in the experiment, are available. When necessary, these variables must be coded as parameters into the DUT model, test bench or the stimuli function. Section 3.1.3 provides more model related hints. These prerequisites enable choosing the response, test case, stimuli, and initial set of factors (i.e. active factors), in this order.

Figure 3.4 shows an example of a DUT model in its test bench. When triggered, the system regulates the voltage of the load. A corresponding DUT configuration file, in a tabular form, to which specification requirements were transfered can be viewed in Figure 3.5.

Such a file is parsed for experiment and simulation configuration. The controller extracts the experiment specific information: number of factors, relative variations of factors and, if existing, expected range for the response, factor distributions and correlations. The Cfg&Stim unit must parse this file as well, and map the factors to parameters of the DUT model and of the test bench,

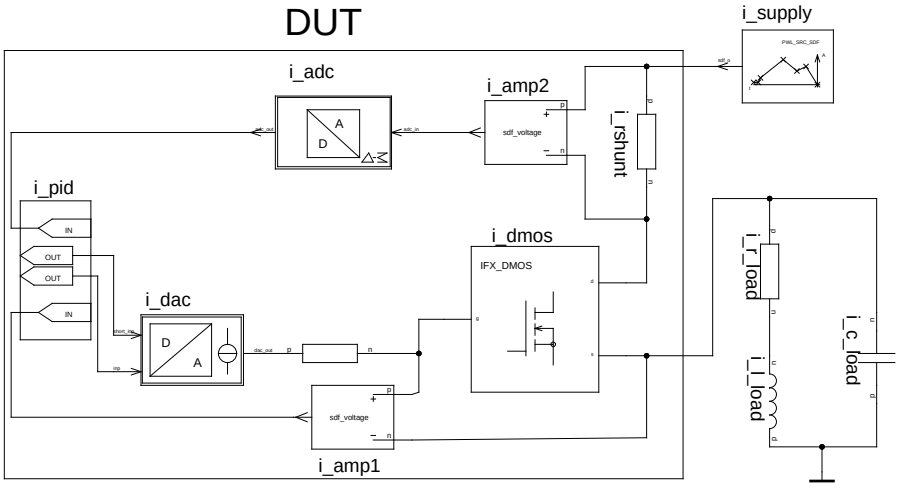


Figure 3.4: Example of a DUT regulating the load voltage

and the response to signals and monitors. This way, it is able to configure them as requested by the controller, prior to each simulation. Other examples of factors and responses, extracted directly from the specification, can be found in the Results chapter 5.

factor	symbol	description	nominal (=cente	tolerance	path in model	active	distribution
	v_source	Supply voltage	...	...	i_supply.p.value	y	uniform
	sh_res	Shunt resistance	...	...	dut->i_rshunt.p.value	y	normal
	L		...	...	i_l_load.p.value	y	normal
	R		...	...	i_r_load.p.value	y	uniform
	C		...	...	i_c_load.p.value	y	normal
	...	...	...	...	...	...	...
response	**name**	**description**	**expected typica**	**expected tole**	**monitor function call**	**measu**	**testcase**
	trigger_delay	delay to supply lo	...	...	meas_delay(current_s	y	trigger
	trigger_sr	trigger slew rate	...	...	meas_sr(current_sf_s	y	trigger
	trigger_crt	trigger current	...	...	meas(current, 50 ms)	y	trigger

Figure 3.5: DUT configuration file

Constraints on how the factors and response are chosen can be modelling concerned, i.e. related to the configurability of the models or to the ability to record and store the simulation outputs. These are detailed in the next section. Alternatively, the constraints can be related to the experiment budget, e.g. maximum number of runs, imposed by the initial requirements, or they can come from the restrictions of the approach: maximum number of factors, width of factor ranges.

3.1.3 Simulation model: requirements and setup

SystemC, with its extensions, is chosen as means of modelling and simulation. It is expressive in terms of modelling capabilities: several models of computation and communication (MoCCs) are available to optimally describe heterogeneous systems. Models are parameterizable, run-time configurable and easily refinable. In terms of simulation, SystemC can achieve a high performance and offers good observability over the run-time behaviour. More reasons for this choice can be found in Section 1.1.3. The following paragraphs detail the requirements and implementation using SystemC for the model and simulation related units.

Model requirements

Some requirements on the models are stated, in order to make use of the above mentioned capabilities. First, the abstraction capabilities must be used and the appropriate MoCC must be selected, so that the trade-off simulation speed versus model accuracy is well controlled. More details on how to control and optimize this trade-off are provided by previous work [RDGP09c, RDGP09a]. Second, the model parameters must be easy to view and configure. This class of reconfigurable model parameters forms an "executable" description of the design in its current state. The values are not hard-coded in the models, rather turned into variables and read-into the model at execution time. They can be accessed when needed, or set to the test case specific defaults when not accessed.

Figure 3.6 shows an example of functional parameters for a switch model, no matter which is the choice as abstraction level or MoC: Electric Networks (ELN) or Discrete Event (DE). Simple control and observability of the current state are enabled by configurable parameters.

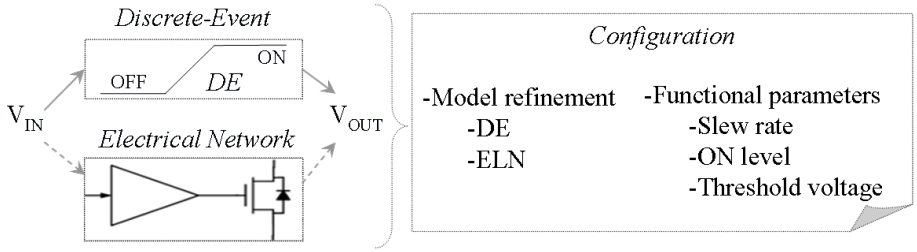


Figure 3.6: Parameterization of a switch model

Third, the model must be easy to inspect i.e. the response must be easy to monitor and record. The response is most of the times a static characteristic of a signal. When only a few time samples of the signal are necessary to compute the response, the related signal should not be traced for the complete duration of the test. Performance-wise, it would be inefficient, because it would cause lower simulation speed and high memory consumption, which amplify for experiments with hundreds of simulation runs. Moreover, it would introduce overhead, in order to postprocess the traced samples, and extract the actual response. The next paragraph introduces the monitors, i.e. functional elements which record static responses. When a transient response is of interest, e.g. to validate a signal over the complete test duration, the tracing should be done with a precision not higher than necessary. Such a transient analysis is presented in Section 4.3.

Monitors

Since SystemC allows inspection in the models during simulation, special monitors are created to track the responses. The requirements on the system usually relate to ranges or extreme values for: voltages, currents, delays, etc. Depending on the nature of the response, monitors can:

- measure signal properties: slew rate, offset, extreme values, settle time, drop in value after a predefined time.
- compare the signal against thresholds, e.g. to record the delay of signal transition to specific values.
- measure delays between signal-related events.

These components are configurable and applicable under different scenarios. The user can provide as inputs the path to the signal to monitor and specific parameters such as threshold value, time/value to start detection, the format or file for the log. The monitor can terminate the simulation once the response has been recorded, to speed up the complete experiment.

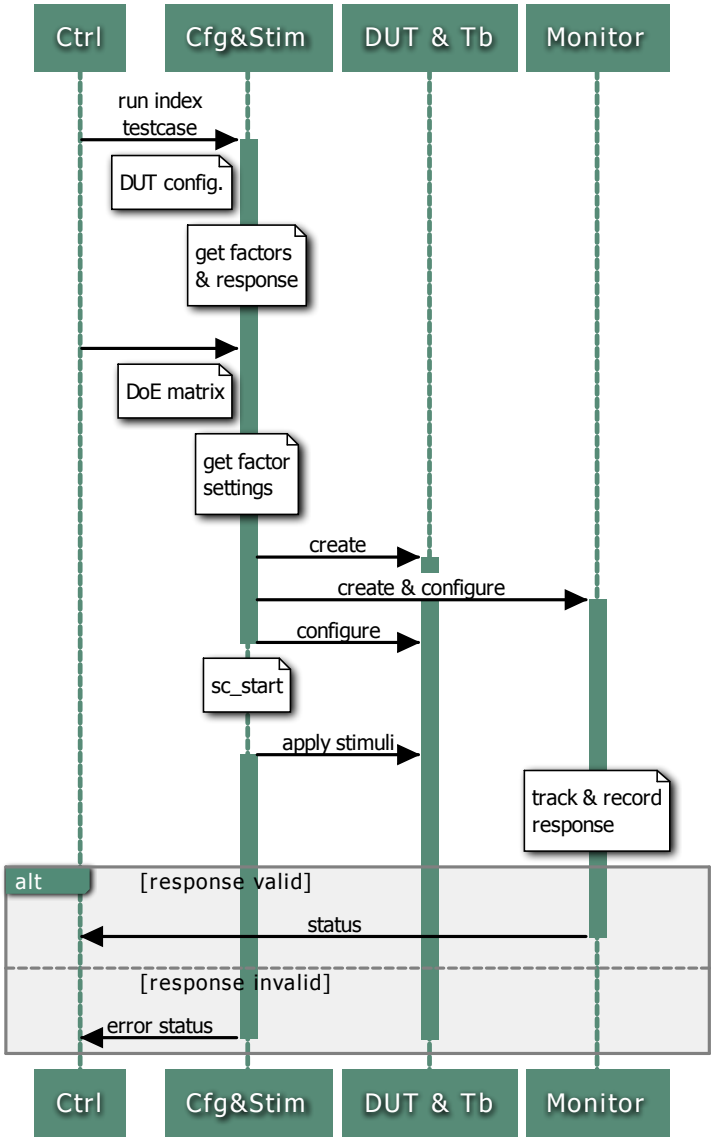
The model is considered valid for the experiment only if the monitor can be provided after the nominal run with the data necessary to compute the response. A response is valid if it is in the expected range, and this is most of the times test case related (e.g. no switch-off delay will be recorded in a test-case which does not switch off). An arbitrary simulation run is valid if a valid model has provided a valid response after the execution.

The Cfg&Stim unit

The Cfg&Stim unit is executed by the controller, using as arguments the test case of choice and the index of the current run, in order to find the corresponding factor settings. The main inputs are the DUT configuration file, and the experiment-specific data in the form of the DoE matrix. This unit includes a configuration part which instantiates, elaborates and configures the DUT, test bench and monitor. The factors and responses are extracted from the DUT configuration file. The rest of the DUT parameters are set to defaults, done either by the configuration part, or by the submodel itself. The second part is responsible for simulation control. Both make use of SystemC constructs. An auxiliary function is also needed for the parsing of the DUT configuration file.

Simulation flow

The flow performed for each simulation, by the Cfg&Stim unit, DUT, test bench and monitor, is visible in the figure.



Simulation flow in SystemC

- Parse the DUT configuration file, to extract:
  - nominal and tolerance values for factors active in the current experiment
  - paths to configure them in the model
  - monitoring function calls (and the path to the signal related to the response)
  - test case associated to each response. This way, responses are tracked only in test cases which can provide them.
- Parse the DoE matrix file, to find the (normed) factor settings for the current run. A check of consistency for the number of factors from the two sources is also performed.
- The test case passed as argument selects the stimuli. The monitors for the test case-related responses are created and configured, as well as the DUT model and the test bench.
- The simulation is executed, by applying the test case specific stimuli.

- After the monitor records a valid response, it returns a proper status to the controller. If an invalid response value or no report from monitor is recorded, a status to indicate simulation failure is returned to the controller by the stimuli function.

It is important to select the type of monitor and create it at run time, because the monitor choice depends on the configuration file, which is written and provided to the Cfg&Stim only at run time. Only after the DUT is created, can the monitor and related signals be created and bound to the model and its test bench. Once these steps are performed, the configuration of the DUT submodules and of the monitor are performed. Some details on this flow from the controller perspective will be given in Section 3.1.5.

3.1.4 Nominal value simulation

The nominal value simulation refers to the center point of the hypercubic factor space i.e. with each factor set to the center value of its range. It has special significance, since it is always the first run, and will be used as a reference for the subsequent simulations. Its results enable checks of functional aspects of the experiment: a failure to run it or to validate that the response is in the expected range means the objective of the experiment must be reconsidered. The user should be involved, mainly to visually assert the functional behavior.

For example, Figure 3.7 shows the nominal value simulation for the system from Figure 3.4, corresponding to the configuration file in Figure 3.5. A visual check of the transient behavior is performed. The values for the responses of interest (also represented in the figure) are checked and stored by the controller, as reference for the experiment.

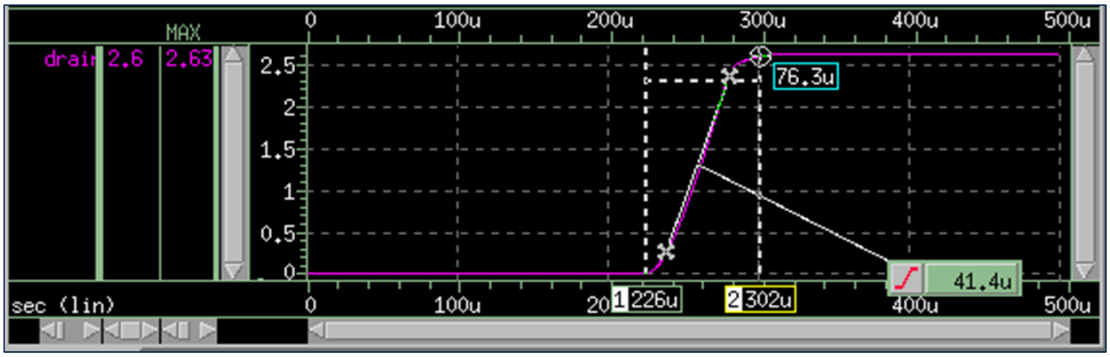


Figure 3.7: Nominal value simulation

During the nominal value simulation, more aspects can be verified: duration of one simulation run, quality of reporting, correct initial settings. These should be detected here, before investing time in the rest of the runs. As noted before, tracing the response-related signal would slow down future simulations and take up memory space, so it will be disabled after this point. More examples of nominal value simulations can be found in the results chapter 5.

3.1.5 Simulation control procedure

The controller acts as master in the complete experiment. It is the interface between the user and the rest of the units of the experiment. The steps it performs are:

1. Prepare the experiment, by
   (a) initial interaction to the user
   (b) parsing of the configuration file
   (c) launching the nominal value simulation
2. Design the experiment (detailed in the following chapter).
3. Control the experiment: once the DoE matrix is generated, each run is launched, by calls of the Cfg&Stim executable, with the test case of choice and the run index passed as arguments. The controller checks the returned status at the end of each execution.
4. Analyze the results, see Chapter 3.3.
5. Report the experiment status and resulting performance.

This flow is modular i.e. these phases can also be performed stand-alone. The process can be extended to an adaptive loop, including a decision-making process dependent on the postprocessing output. This is covered in Section 4.1. The extension can also be performed in order to analyze transient responses, as shown in Section 4.3

Similarly to the nominal simulation, the status and the response are checked for each run. For each failed run e.g. failed status returned, no result recorded, or a response out of range, a decision to abort the experiment can be taken. Alternatively, "Not-a-Number"s (NaNs) can replace the failed runs, to be ignored by the analysis part. This way, the failed simulation issue does not escalate e.g. response values with wrong orders of magnitude are detected in good time, and offsets in the matrix of results caused by missing responses are avoided. Otherwise, subsequent postprocessing steps would be misleading, and would compromise the complete experiment.

MATLAB is chosen for the implementation of the controller. It can easily control the SystemC simulation and its toolboxes help to implement the algorithms described in the next chapters. MATLAB also facilitates handling, processing and visualization of large data structures, which is necessary for the experiments under study.

The data flow i.e. the transfer of configuration data to the simulator, and of simulation results to the controller, is implemented by simple file intercommunication. The simulation is executed by the controller by simple system calls $system('exec\_name\ arg1\ arg2...')$. The command line arguments can pass any necessary parameters, while using environment variables was found to be a lower-performance alternative.

3.2 Designing the experiment

Figure 3.8 represents the main inputs and outputs of the experimental design step. This step outputs the DoE matrix, with factor settings for each run.

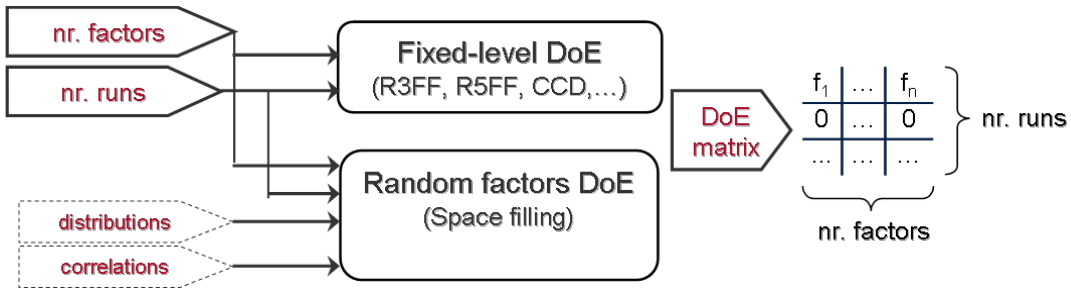


Figure 3.8: Experiment design flow

The general goal of experimental design is that the postprocessing of results can provide maximum information about the response after minimum number of runs. Further targets are:

- ability to handle maximum number of factors
- minimum initial assumptions on the response behavior over the verification space
- accuracy in predicting the response, with a special focus on the worst-case

Section 3.2.1 presents principles of the classical DoE found valid in simulation experiments on the systems under study. Assumptions made about the response, decided by a metamodel choice, and their impact on sensitivity analysis, are discussed in Section 3.2.2. Section 3.2.3 translates the requirements on factors in terms of levels and distributions, into inputs to the experiment planning step. Fixed-level DoEs, presented in Section 3.2.4, use a fixed set of levels for the factors and build optimal DoEs with minimum number of runs. Alternatively, random factor DoEs (RFDoEs), described in Section 3.2.5, invest more runs, but provide more insight in the factor space, by distributing the factors according to their statistical properties.

In order to plan effective experiments, the approach contributes with deciding which principles, metamodels and classical DoEs apply to the simulated systems. Given the requirements and the budget constraints, as well as the nature of the factors and of the simulation model, the selection of experiments is validated on deterministic responses, before being applied to simulation cases. For the experiments with random factors, the present work must contribute by generating the proper distributions and correlations for factors, and evaluating the coverage such DoEs reach.

3.2.1 Principles in the design of simulated experiments

This section describes the core principles of experimenting. They are either directly adopted from real life experiments, or adapted to simulated systems. The general problem of experimental design in the presented context is to plan simulation sets in order to extract an optimal approximation of the response. A design is identified by the DoE matrix d of factor settings for the experiment. The analysis of the vector of results r must find the best estimate for the coefficient vector c so that:

$$res((R(c,d),r)) = min. \tag{3.1}$$

where:

- R is the metamodel which approximates the response.
- c is the set of parameters (or coefficients) which must be estimated.
- d is the DoE matrix of factor values used in the experiment. It is an $m \times n$ matrix, m is the number of runs and n is the number of factors.
- r is the vector of length m of response results, corresponding to the DoE matrix d.
- res is the residual function, i.e. a metric of the distance between the results r and the estimates given by R.

The type of R and the number of runs m influence the DoE choice, which in turn determines the d matrix.

The experiment objective The steps common to any experiment are followed, as introduced in Section 2.3.2. The first step is to define the objective of the experiment. Screening, sensitivity analysis and worst-case analysis are sequentially addressed here, with respect to common objectives of metamodelling methods. The next step, i.e. definition of the responses and factors, is derived from the initial requirements, as described in Section 3.1.2, and with the practical restrictions explained in Section 3.1.3.

Response assumptions The sparsity of effects principle states that the system is likely to be driven primarily by some of the main effects and low-order interactions. This means it can be assumed that some of the factors have insignificant impact on the response, as compared to the rest. The low-order effects prevail over the higher order ones and from an order upwards, the effects can be neglected. These assumptions often apply in practice, and can be supported by a choice of small factor ranges, e.g. a relative variation of less than 50%. Factor ranges must be small enough anyway for the purpose of preserving the functionality i.e. avoid unpredictable factor effects and a corresponding misbehaviour of the system. Such small ranges ensure low-order effects, and keep the response variance under control.

These assumptions are the starting point when choosing the metamodel and the corresponding experimental designs. The metamodelling Section 3.3 introduces methods to evaluate the correctness of estimates. Classical principles recommend a sequential approach, which invests no more than 25% of the total number of runs in the first experiment. A sequential approximation strategy should be deployed, which experiments and refines the estimates until the metamodel is fit. Section 4.1 automates such a self-correcting flow, which revises the assumptions which are found not valid.

Factor controllability An important advantage of the simulated systems is that all factors are controllable. Noise factors are not considered, and neither are their consequences on the experiment planning and analysis. As compared to classical experiments, another benefit of controllability is that, at least in theory, any factor level can be used. The consequence of driving the system into a state not reachable in reality i.e. corresponding to unrealistic factor settings should be detected by an invalid response value.

Discretized factor ranges For each factor, the sampling algorithm selects out of the range of possible values, be it continuous or discrete, a discrete subset for the specific experimental design. Then, the analysis interpolates over the factors' ranges. When discretizing the ranges, the sampling algorithm chooses either predefined factor levels, by fixed-level DoEs (Section 3.2.4) or random levels distributed in the range (Section 3.2.5).

No levels outside factor ranges In some experimental designs e.g. Central Composite Designs, the levels could fall outside the factor's range. Similarly, the tails of normal distributions can be outside the region under study. In such cases, there is no guarantee that the system can even provide a response. Therefore, the reachable factor levels used here are restricted to within the configured ranges. These limit the verification space and the response approximation is used only inside these ranges.

No replication Replication, i.e. rerunning tests with the same factor settings, is used by classical experiments to estimate the influence of uncontrollable factors such as the measurement error and to obtain a more precise estimate of the mean response. It can highlight the sources of variability both between runs and within runs [Mon05]. In our case, however, sample points must not be resimulated, because more runs with control over all factors would clearly yield the same response value. Still, replication of their results only can be useful for the analysis, to apply methods of the classical DoE, e.g. to weigh the specific points more in the metamodel estimation. The experimental error will be estimated null, while the remaining response variance is attributed to the factors or to the metamodel unfitness.

No randomization, no blocking Randomization is necessary in real life cases and implies that the experimental material and the order in which individual runs are performed should be randomly determined [Mon05]. This way, the impact of measurement conditions, such as human errors or the environment, must be homogeneous. Blocking means grouping the runs into batches, which have relatively homogeneous experimental conditions. It is used to eliminate by compensation the influences of uncontrollable factors e.g. noise [Mon05]. Since these influences do not impact simulation results, none of these principles nor their consequences are applied here.

Further issues when adapting the theories of experimental design to simulations are related to the metamodel representation, which is presented next.

3.2.2 Effects of factors and response metamodels

Metamodels are empirical models which approximate the response as a multivariable function on the factors. A form is assumed, which is fitted by simulation data. The general approach to fitting it is referred to as regression analysis, while the choice of simulation data to fit various metamodels is referred to as experimental design.

It is of interest to choose intuitive metamodels which highlight factor effects, i.e. the impact of factors on the response can be easily extracted. That is why parameterized metamodels are chosen, and their coefficients are used to characterize the factor set. At least one quantity should be associated with each factor as this allows comparison and interpretation. Therefore, the main requirements for a metamodel are:

- small number of runs, i.e. little data necessary to estimate it
- simple fitting algorithms
- possibility to extract factor effects out of the metamodel coefficients

Multivariate polynomials are regression metamodels which meet these requirements, as long as they pass the fitness tests, described in Section 3.3.2.1. Their general form, for an n-dimensional factor space, is:

$$R = c_0 + \sum_{k=1}^{o} \sum_{i=1}^{n} c_i^{(k)} \cdot F(i)^k + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} \cdot F(i) \cdot F(j) + ... + c_{i...n} \cdot \prod_{i=1}^{n} \cdot F(i) \tag{3.2}$$

Here, $F(i)$ is element i of the vector of factors F of length n. Such metamodels can be decomposed into components, to decouple factor effects of different types. Coefficients quantify these effects:

- c_i: individual factor effects: linear or main effects for $k = 1$; quadratic effects for $k = 2$.
- $c_{i...k}$: interaction effects; c_{ij} are the 2-factor interaction effects.

The statistical model includes the constant term, n individual effects for each order, $C_2^n = n \cdot (n-1)/2$ 2-factor interaction effects, C_3^n 3-factor interactions etc. and one n-factor interaction, so there are 2^{n-1} effects. The low-order effects, i.e. main, 2-factor interaction and quadratic effects sum up to $(n+1) \cdot (n+2)/2$.

Figure 3.9 shows these basic factor effects, in a 2-dimensional factor space. The response surfaces correspond to metamodels with only one of these components.

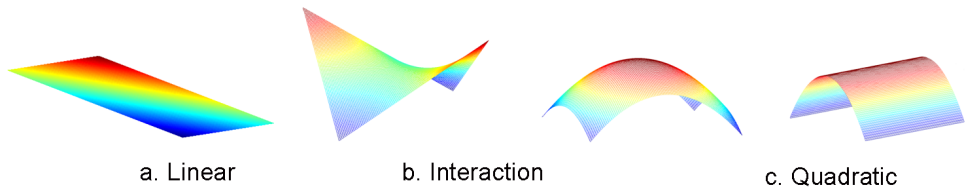


Figure 3.9: Basic factor effects

Projected on one factor dimension, these effects are visible in Figure 3.10. The interaction effect determines an impact of one factor on the response which depends on the level of the other factor.

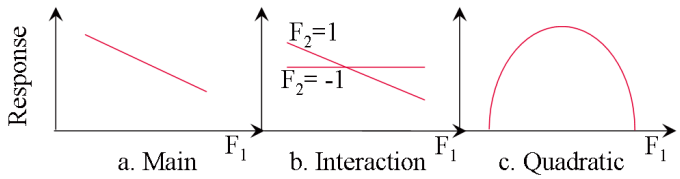


Figure 3.10: Basic factor effects in one dimension

Such types of low-order effects are quite common in practice:

- linear effects e.g. $end - to - enddelay(block_i delay) = \sum_i block_i delay$
- 2-factor interaction effects e.g. derived from Ohm's law $U(R, I) = R \cdot I$
- quadratic effects in resonant systems e.g. $Fourier(U)(frecv) = f(frecv^2)$

The sparsity of effects principle implies that significant effects come only from a reduced factor set and are of low-orders. Higher order factor effects rarely occur, but when they do, an iterative estimation of the next order effect is possible, as the method introduced in Section 3.3.3 demonstrates. The sequential strategy proposed in Section 4.1, starts from here and realizes a refinement of the polynomial metamodel. After the metamodel is built, the sensitivity analysis is addressed, by comparing factor effects against predefined thresholds and to each other (Section 3.4.1).

To summarize, multivariate polynomials are good choices as regression models, because they:

- Do not need complex or simulation-intensive experiments (Section 3.2.4).
- Are easy to build i.e. require simple regression algorithms (Section 3.3.2.1).
- Are easy to interpret (Section 3.4).
- Can handle enough factors (Figure 3.15).
- Can be deterministically optimized for worst-case analysis (Section 3.4.3).

More about metamodelling techniques can be found in Section 2.3.1.

3.2.3 Levels, probability distributions and correlations in factors

Each factor is characterized by the levels it can take during the experiment, its probability distribution as well as the correlations to other factors. They are provided by the initial requirements on the system and each should be accounted when designing a proper experiment.

Levels

A factor's levels are the values it takes during the experiment. A benefit of simulated experiments is that there is no practical difficulty in using extra factor levels. The accuracy of results is not affected either. A higher number of levels per factor can involve extra computational complexity when estimating the metamodel or deriving factor effects, but has potential to increase the accuracy of the estimates.

Depending on the experiment type, 2, 3, 5 or more levels per factor are used. These are distributed between -1 and + 1, equidistant, equally distanced from the center, or randomly. The levels are decided by the DoE choice: fixed-level DoEs usually involve up to 5 levels per factor, while RFDoEs use a different level in each run, dependent on the factor's statistical distribution.

Factors are normed to their ranges, so that the verification space becomes a hypercube of radius one[1]. Using normed values makes the planning and analysis of results for each experiment easier to implement. E.g. comparisons between factor effects becomes easier when using normed values. Moreover, they contribute to decoupling between the controller of the experiment, which must be aware only of the normed values, and the DUT dependent part, which has to use the complete factor ranges.

[1]Factor correlations reduce the hypercube to the subregion which complies with the existing constraints

Probability distributions

Probability distributions of factors should be considered when they are available and when sufficient runs can be invested, i.e. in RFDoEs. The required number of runs is discussed in Section 3.2.5. Factor effects and the response distribution can be extracted as influenced by the statistical properties of factors. This type of sensitivity analysis is introduced in Section 3.4.4, along with a derived worst-case study.

The probability density function (PDF) of a random variable gives the probability for the variable to take specific values. The probability that the variable falls within a given range is given by the integral of its density over the set. The cumulative distribution function(CDF) describes the probability that the random variable is found at a value less than its argument. Figure 3.11 presents histograms of factor samples, for some of the statistical distributions of interest. The PDF is used as a reference for the generated values as the red lines indicate in some of the examples.

- Corner-peaked
- Uniform
- Maximum-peaked
- Gaussian
- Minimum-peaked
- Piecewise linear cumulative distribution function (PWL CDF) i.e. piecewise constant probability distribution function (PDF)
- Piecewise constant CDF i.e. discrete PDF

The distributions of interest are generated using the statistical functions available with MATLAB [52], modified when necessary. Others are derived from the statistical package implemented for SystemC, introduced by [MHE08]. More examples of PDFs of interest can be found in Appendix B.4.

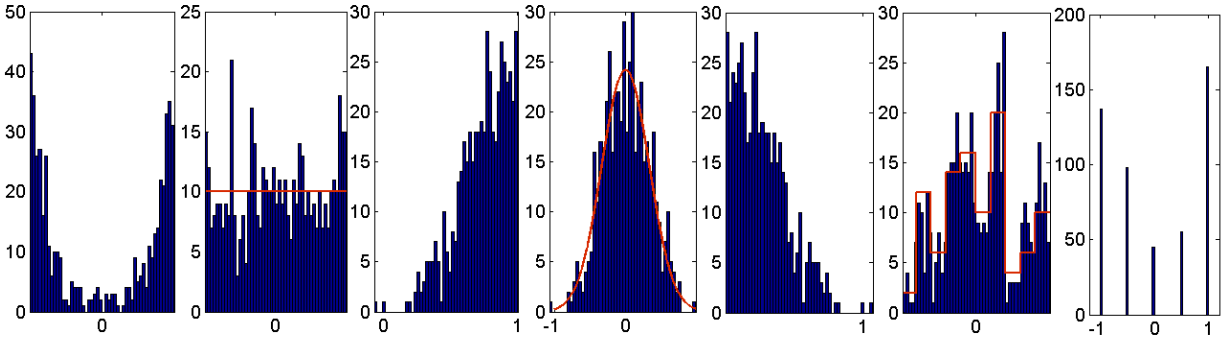


Figure 3.11: Factor statistical distributions

In a reverse manner to how these reference PDFs are used to generate samples, the PDF parameters can be estimated to best fit an input data set. The simulation results of the response are analyzed by such a method, detailed in Section 3.4.4.

Correlations

2-factor correlations represent the codependency between pairs of factors. Such codependencies frequently occur in practice, when factors originate from common sources of variation, e.g. temperature can generate correlated factors in system components. Accounting correlations into the experimental design introduces a higher coverage of the factor space, with respect to the real distribution of samples. Such correlations best suit RFDoEs, where no hard restrictions on fixed factor levels or on the design orthogonality are imposed, but can be implemented in the case of fixed-level designs, as well. The sets of values used for codependent factors are correlated to each other, which transforms the initial hypercubic factor space into a non-uniformly covered space.

Factor correlations can be generated in more ways:

- Specifying factors as any deterministic function on a subset of factors e.g. linear combinations of other factors e.g. $F_2 = \alpha \cdot F_1$. This can be implemented in any experimental design and makes the analysis of results simpler, when a metamodel must be estimated.
- Linear correlation, using the Pearson correlation coefficient [58], which is mainly sensitive to a linear relationship between two variables ($-1 \leq corr(F_1, F_2) = ct. \leq 1$).
- Computing the values of one factor out of other factors and explicitly introducing an extra source of uncertainty. Example: $F_2 = \alpha_1 \cdot F_1 + \epsilon$

The last two correlation types are more appropriate for the RFDoEs. Figure 3.12 shows samples in a 3-dimensional space, where the factors are normally distributed and linearly correlated. A correlation matrix, marked on the plot, is used to specify the Pearson correlation in factors. The plot matrix shows on the main diagonal histograms of the samples, corresponding to normal probability distributions, while the other subplots represent the factors against each other.

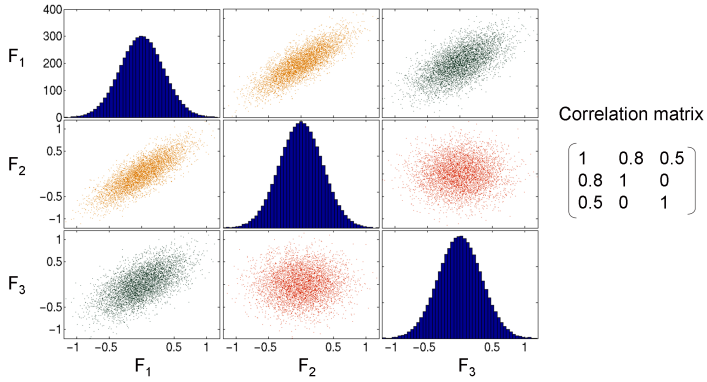


Figure 3.12: Factor correlations

Figure 3.13 plots 2-factor correlations. These are weaker on the left side, where both F_1 and an extra source ϵ influence F_2. The right side shows a strong correlation where F_2 is determined by F_1 only.

For the regression necessary to build a metamodel, when factors are independent, direct regression is more likely to result into valid estimates. Therefore, the analysis of results must decouple the factors into independent sources of variation. Only then can the variance in response truly be

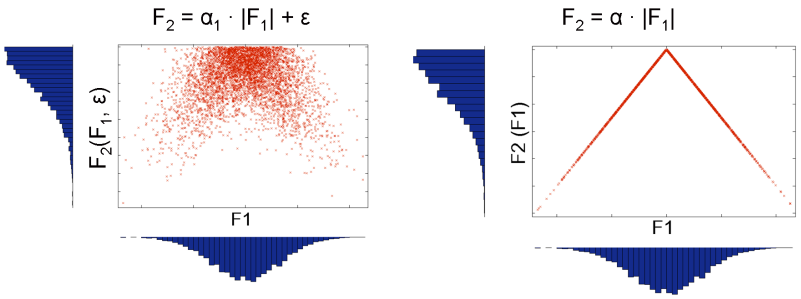


Figure 3.13: 2-factor correlations: weak vs. strong

attributed to the variations applied in factors. Section 3.3 describes how to decouple correlations when analyzing the results.

The response is characterized as a result of the experiment, thus indirectly accounts for these correlations. In a similar manner to how correlations can be introduced in factors, they can be extracted from the set of response values in relation to the sets of factor values. This is a form of sensitivity analysis, when effects of factors are quantified by the correlations to the response. More on this topic is discussed in 3.4.4.

3.2.4 Selected experimental designs with fixed-level factors

Experiments with fixed-level factors use for each factor one of the predefined levels. They select a reduced subset of runs, out of the set of $number\_of\_levels^{number\_of\_factors}$ possible factor combinations. A DoE should be chosen only after defining the metamodel type to fit. This way, response assumptions are consciously decided on, in order to be aware of the limitations and of the possible experimental follow-ups. The metamodel does not impose hard restrictions on the DoE choice, but a lower chance of errors is ensured when the DoE is built as to optimally fit the chosen metamodel. The classical, fixed-level DoEs estimate coefficients of polynomial regression models, optimizing the number of runs with respect to the coefficients to extract. This is in fact the main merit of the classical DoE, for the present approach.

DoE properties which minimize the risk of errors in the metamodel fitting step, are:

- orthogonality: a design is orthogonal when the DoE matrix has pairwise orthogonal columns: $d(:,i) \cdot d(:,j)' = 0$. In this case, $d' \cdot d$ is a multiple of the identity matrix. Orthogonality reduces the errors in the coefficients estimated after regression, therefore allows independent analysis of the considered factor effects. This is justified in Section 3.3.2.1.
- rotability, i.e. equal precision of estimation in points equally distanced from the center: is achieved when the DoE points are located on hyperspheres centered on 0. For instance, any first order orthogonal design is rotable [Mon05, p.282-296].
- minimum response variance: minimum probability of prediction errors. It can be achieved for a first order model when $\sum_{k=1}^{number\_runs} d(k,i)^2 = constant, \forall\ i = 1 \ldots\ number\_factors$ and the DoE is rotable.

- equality of factor treatments: although normed factors have the same ranges, i.e. $[-1...1]$, their unnormed initial ranges are different. The more a factor varies relative to its nominal value, the more complex effects it is probable to have. Therefore, more coefficients related to such factors should be attributed. More levels and more runs in which the factor is varied should exist. Such a situation can be handled with the custom DoE presented in Section 3.2.4.1.

A consequence of the sparsity of effects principle is that the worst-case has a higher probability of being located in a corner, because the main and 2-factor interaction effects are more likely to occur than higher-order effects. Therefore, a fixed-level design should minimize the prediction error in the corners. This happens when most of the DoE points are also located in the corners and the design is approximately rotable.

3.2.4.1 2-level fractional factorial designs

Use 2 levels per factor. A 2-level full factorial design uses all combinations of levels and it can estimate a regression model with all main effects and interactions of all orders. But such a metamodel includes interactions of unnecessary high orders. It is not desired because it determines an exponential increase of number of runs with the number of factors, just as exhaustive search methods do. Therefore, 2-level fractional factorials are preferred. In the following, concepts of 2-level fractional factorials are selected from textbooks such as [Mon05, p.314-318] and clarified.

A 2^{k-p} fractional factorial design contains a fraction of the full 2^k factorial of 2^{k-p} runs. The fractional factorial DoEs are heavily used for early design and improvement, in screening experiments, and for later response characterization. Their main advantage is that they require only a few runs to estimate a predefined subset of the full set of metamodel coefficients. When the metamodel is inadequate, the initial fraction can be augmented with a further fraction to separate effects initially assumed insignificant from the ones assumed significant.

Definitions

A fractional factorial has a resolution R when there is no main effect aliased with an effect of an order smaller than $R-1$ (exclusively), no 2^{nd} order effect is aliased with an effect of order smaller than $R-2$... [Mon05]. A resolution 3 fractional factorial (R3FF) estimates main effects aliased on 2-factor interactions. Such a DoE is commonly used for screening, because it can extract preliminary main effects. A resolution 5 fractional factorial (R5FF) can decouple main effects from 2-factor interactions, but 2-factor interactions are aliased with 3-factor interactions and so on.

To build the DoE matrix of a fractional factorial, the full factorial in $(k-p)$ factors is first written. + and - symbolize 1 and -1 normed factor levels. Then, the columns for the rest of p factors are written as to be identical to the columns of appropriately chosen interactions involving the first $k-p$ factors. Such interaction columns are obtained by multiplying the signs of the respective factors. The selected p interactions are called the design generators. All columns equal to the identity column i.e. only with + signs, form the defining relation for the design. The design resolution is equal to the smallest number of letters in the defining relation. The example Table 3.2 illustrates these concepts.

The loss when fractioning the full factorial to allocate less runs is that the estimated effects are aliased (overlapped) with the effects initially assumed not to exist. Multiplying any of the design generators yields the generalized interactions, i.e. words. The aliases of any effect are produced by the multiplication of the column for that effect by each word in the defining relation. The generators must be chosen so that potentially important effects are not aliased with each other. When a more complex metamodel should have been chosen, the initially estimated metamodel accounts in its coefficients (additively) for the extra coefficients of the better fitting metamodel. To solve this, a dealiasing experiment must follow, which augments the existent fraction with another one.

Properties

- 2-level fractional factorials have the rotability property previously defined and are orthogonal.
- Projection property: the 2^{k-p} design collapses into a full factorial or a fractional factorial in any subset of the $r \leq (k-p)$ of the original factors.
- Fold over of R3FF to separate aliased effects: by combining fractions in which certain signs are reversed, effects of potential interest can be isolated. This property is useful in sequential experimentation.

Table 3.1 includes in its cells the number of factors which can be handled with fractional factorials with the number of runs included in the title row.

Number of runs	*4*	*8*	*16*	*32*
Design type	*Number of factors*			
Full factorial	2	3	4	5
Half fraction	3	4	5	6
Resolution IV fraction	-	4	6-8	7-16
Resolution III fraction	3	5-7	9-15	17-31

Table 3.1: Number of factors versus number of runs in fractional factorials
source: [Mon05], Table 8-28

Software packages such as [52] are used to generate the desired fractions and their alias structure, which can be quite complex, especially when the number of factors is high. Appendix B.2 includes the resolution, number of runs and design generators for selected fractional factorial designs, for a number of factors up to 15 and a number of runs of up to 128.

Example: building the resolution 5 fractional factorial for 5 factors

A table of treatment combinations (Table 3.2) is formed which shows by + and - the settings of each factor and of the corresponding interactions, for each run. To build this table for the highest resolution fractional factorial, i.e. resolution 5, the basic design for the full 2^{5-1} factorial is written. Then the 5^{th} factor is added by identifying its plus and minus levels with the ones of the highest order interaction, i.e. $ABCD$. $ABCD$ is the design generator and $I = ABCDE$ is the defining relation. The alternate fraction is obtained using the generator $I = -ABCD$.

	Full 2^4 *factorial*				
run	A	B	C	D	$E = ABCD$
1	-	-	-	-	+
2	-	-	-	+	-
3	-	-	+	-	-
4	-	-	+	+	+
5	-	+	-	-	-
6	-	+	-	+	+
7	-	+	+	-	+
8	-	+	+	+	-
9	+	-	-	-	-
10	+	-	-	+	+
11	+	-	+	-	+
12	+	-	+	+	-
13	+	+	-	-	+
14	+	+	-	+	-
15	+	+	+	-	-
16	+	+	+	+	+

Table 3.2: The one half fraction of the 5-factor factorial

Factor effects are extracted by adding the treatment combinations where the factor is set to + and subtracting the ones where it is set to -, e.g.
$effect_A = -\sum_{i=1}^{8} response\_run_i + \sum_{i=9}^{16} response\_run_i$.

Since some of the interactions are obtained in the same way, it is impossible to differentiate between these interactions and the respective main effects. In fact, what is being estimated is the sum between the effects of interest and their aliases. The alias structure of the design is easily extracted from its defining relation $I = ABCDE$. For instance, $A \cdot I = A \cdot ABCDE = A^2 \cdot BCDE = BCDE$. Each main effect is aliased with a single 4-factor interaction. The alias structures of the 2-factor interactions are:
$AB = CDE$; $AC = BDE$; $AD = BCE$; $AE = BCD$; $BC = ADE$; $BD = ACE$; $BE = ACD$; $CD = ABE$; $CE = ABD$; $DE = ABC$.

Response surface designs

Use at least one more level per factor, to estimate quadratic factor effects, in number of $2 \cdot n$. A Central Composite DoE (CCD DoE) is a 2-level factorial design augmented by center points and 2 star points (i.e. axial runs) for each factor. The star points visible in Figure 3.14c are placed on the axes of the hypercubic factor space, where one factor has the value $\pm\alpha$ and all others are 0. The value of α which is used is 1 (the design is a face-centered CCD). The design only uses 3 levels per factor and it is not rotable (the region of interest is cuboidal). The option of $\alpha = \sqrt{factor\ number}$ (circumscribed CCD) would make it approximately rotable and would involve 2 additional levels per factor. It is not used because the levels would fall outside the factor ranges. Other choices for α are possible depending where the response variance should be minimized.

The center point is replicated in the matrix of results only when it is necessary, e.g. when the regression must weigh the center point more than others. Alternatively, a Box-Behnken DoE [52]

also uses 3 levels per factor, but the points are mostly placed on the sides of the factor hypercube. The two direct disadvantages are:

- The fractional factorial DoEs, consisting of corners, are not subsets of this response surface design, therefore the runs cannot be reused in sequential designs.
- The minimum response variance is not in the corners, because the sample points are not placed in the corners. This violates the condition stated in the properties at the beginning of the section.

Figure 3.14 represents DoEs for 3 factors: the full factorial, the half fraction (R3FF) and the CCD.

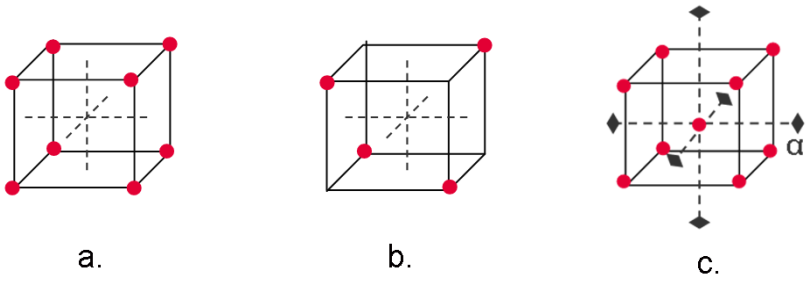


Figure 3.14: Basic DoEs
a. 2^3 Full Factorial b. 2^{3-1} Fractional Factorial c. Central Composite DoE

Figure 3.15 shows the number of runs required by such DoEs. The R3FF is involved for the initial phase of factor screening, and to extract a linear metamodel (only main effects). The R4FF, R5FF and CCD DoEs are used to build metamodels of 1^{st} and 2^{nd} orders. The DoEs are generated using the MATLAB fraction generator i.e. the function $fracfact$.

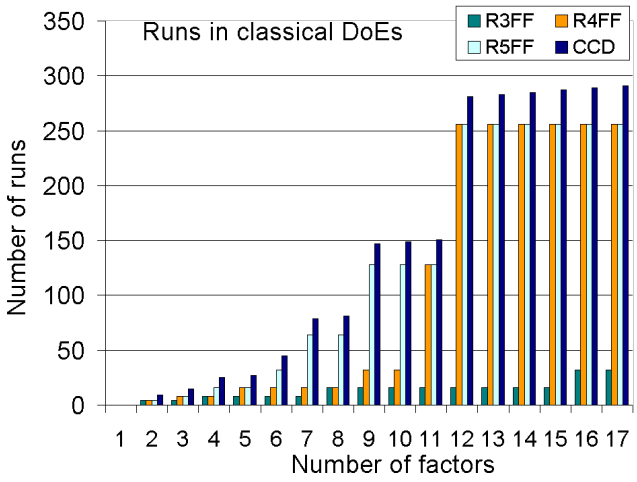


Figure 3.15: Number of runs in fixed-level DoEs

These are small sized DoEs needed to efficiently fit the low-order metamodels. They are part of the classical DoE, adapted for the issues addressed here, e.g. by not using factor levels outside the

[-1...1] range. The approach also needs to be able to estimate higher-order individual effects. To estimate a factor's next order effect, the previous designs are augmented with axial runs for the respective factor. This is deployed in the iterative regression strategy, described in Section 3.3.3. E.g. sweeping a factor's range with k points can estimate individual effects up to the $k-1_{th}$ order. Additional runs can be necessary for the following reasons:

- The nominal run is needed as a reference for the experiment.
- Pilot runs, e.g. sweeping factors, to give a first impression on the response variance and magnitude of effects.
- To validate the metamodel (measure the prediction error of the metamodel) against runs different from the ones used to fit it (as explained in Section 3.3.2.2).
- To optimize the estimate for the metamodel's coefficients (Section 3.3.3).

Sources in the DoE literature indicate to many more experimental designs [SPKA01, WS07], which are, however, not necessary here. Rather simple DoEs can estimate simple metamodels, which are found sufficient, and they involve minimum runs for that. Then, worst-case analysis can focus on the regions of extreme response values, e.g. by optimizing the existing metamodels.

Custom DoE

It is a multi-level factorial design, built to address the present problems. The work presented in [RDGP09b] shows results of such custom DoEs. The design favors a so-called main factor, by sweeping it over several fixed levels in its range e.g. 10 levels. It uses random or fixed levels for the rest of factors, referred to as the factor set. The factors which present particular interest to the experiment can be set as main factors in such a custom DoE. For instance, a main factor can be a factor which has a relatively wide initial range or a factor which determines right from the start a big response variance, identified from a screening phase. Another example is a factor with high interaction coefficients, estimated after a proper experiment, e.g. a R5FF.

A custom analysis of effects for these DoEs is introduced in Section 3.4.1. This analysis focuses on the interaction between the main factor and the factor set. In addition, a piece-wise linear main factor effect can be approximated, to detect non-monotonic, or even discontinuous dependencies of the response on the factors. Such effects determine worst-cases corresponding to main factor values which are not corners, and which depend on the factor set because of the interactions.

3.2.5 Space filling designs and DoEs with random factors

This section introduces experimental designs with random factors. While the concept of space filling designs is well-known, the evaluation of the coverage and of the correlation, as well as the extension to RFDoEs are realized to address the present problems.

Space filling designs

A space filling DoE strategy should "evenly" cover the entire factor space. Several space-filling criteria are discussed in the literature, e.g. maximin, minimax, IMSE, and maximum entropy [WS07]. To aim for a better and more uniform coverage, the distance between sample

points of the DoE is optimized. More precisely, the minimal distance between pairs of points must is maximal (e.g. maximin designs [57]). Different distance metrics can be used, out of which the euclidean one is considered sufficient.

The Latin Hypercube Design (LHS DoE) generates samples in order to ensure that each region of the space is equally represented. Each factor level occurs only once, so DoE points do not share coordinate values. They do not replicate or group points like classical experiments sometimes do [57]. E.g. to sample m points in an n dimensional space, LHS perform the following steps:

1. Divide the interval of each dimension into m non-overlapping intervals having equal probability (for uniform distribution, the intervals have equal size).
2. Sample randomly from a uniform distribution a point in each interval in each dimension.
3. Pair randomly (equal likely combinations) the point from each dimension.

These designs can be generated using the MATLAB *lhsdesign* function. The function can maximize the minimum euclidean distance between pairs of sample points. Figure 3.16 represents the number of runs versus number of factors obtained for various distances between points. Although more runs than in fixed-level DoEs are needed, the number can be controlled and limited, by using the target distance between points.

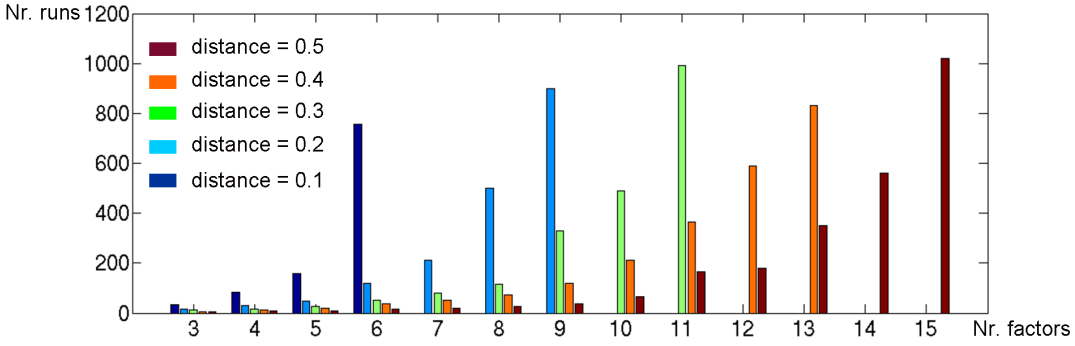


Figure 3.16: Number of runs in LHS designs with uniformly distributed factors

Sources like [Kle08] are more optimistic when estimating this number of runs: a linear increase of the number of runs with the number of factors is present, which does not significantly exceed 200 runs, even for up to 30 factors.

Random factor DoEs

Random factors DoEs (RFDoEs) distribute the factor values according to their properties. Generators of several distribution functions can be derived from statistical packages like [50] or [MHE08], no matter the size of the factor space. Various distributions can be implemented, while accounting for factor correlations, as exemplified in Section 3.2.3.

The main differences between RFDoEs, as defined here, and Monte-Carlo simulations are explained in the following. With respect to planning the simulations, Monte-Carlo methods randomize underlying device parameters, usually normally. RFDoEs, on the other hand, have more power, because they cover a wider class of functions and can introduce correlations. The outputs

of this process are also fed to the postprocessing part. Concerning the results' analysis, Monte-Carlo involves no postprocessing, except for a graphical analysis and an extraction of the worst simulated case. RFDoEs, on the other hand, quantify factor effects in correlation coefficients, and characterize the response distribution. Then, the worst-case can be estimated after computations on this extracted statistical function.

RFDoEs can also perform regression on results to estimate a metamodel, and continue with the sensitivity and worst-case analysis, just as in the case of the fixed-level DoEs. The accuracy of estimates depends on the number of runs, but also on the factor correlation. Results become "biased" by the initial distributions, having more accurate estimates in the centers of distribution e.g. around nominal values in the case of a normal distribution. These steps are detailed in the analysis of results and response characterization parts, in Section 3.4.4.

When combined with space filling strategies previously introduced, RFDoEs can be optimized and evaluated in terms of coverage. The minimum distance between points is maximized by a LHS design. Its value is used here to measure the coverage of the DoE. The numbers in Figure 3.16, although extracted for the case of uniformly distributed factors, can be generally used as reference for space filling and RFDoEs. Since they ensure the target coverage in the uniformly distributed case, then in the presence of differently distributed, and possibly correlated factors, the distance value changes, but the coverage is kept. This is because the shape itself and the size of the verification space also change. This way, when the target coverage is given by a fixed distance, the number of factors is taken into account as the sole impact on the number of runs.

When regression is performed on the results, optimal estimates are extracted when the design is as close as possible to an orthogonal design, similarly to fixed-level factor DoEs. The correlation matrix of orthogonal DoEs is the identity matrix. In order to measure of the performance of an experiment from this point of view, the factors must have a correlation coefficient as close as possible to 0. Therefore, space filling designs should have the correlation between the columns of the DoE matrix smaller than a predefined threshold. The 0.05 threshold for the coefficients of factor correlation is used to conclude whether a space-filling design is good enough from this point of view. This is considered the highest value which allows correct regression on deterministic responses. The correlated ones should have a coefficient close enough to the one initially specified. Together with the distance between points and with the accuracy of representation for the statistical properties, the correlation represents a measure of performance for the step of design of the experiment.

Implementation algorithm The generation of the DoE matrix as described before is performed by the function get_doe_matrix, with the signature:

```
doe = get_doe_matrix(factor_nr, max_run_nr, doe_type, factor_stat_struct)
```

The main inputs are the number of factors, maximum number of runs and whether the DoE is fixed-level or a random factor experiment. The structure of factor distributions and correlations can optionally be passed, when RFDoEs are generated. A matrix of effects could also be specified, for the factorial generator to return a proper fraction of the design. Such a matrix marks by ones the position of effects to be extracted. A pseudo-code description of it is inserted in the Appendix A.1.

3.3 Building the metamodel

This chapter describes the postsimulation steps which form a metamodel accurate enough to approximate the response. Starting from the results of the planned DoE, be it a fixed-level or an RFDoE, parameters of the metamodel are estimated, evaluated for fitness and optimized. Figure 3.17 represents the main inputs and outputs of this step. The dotted steps are optional[2].

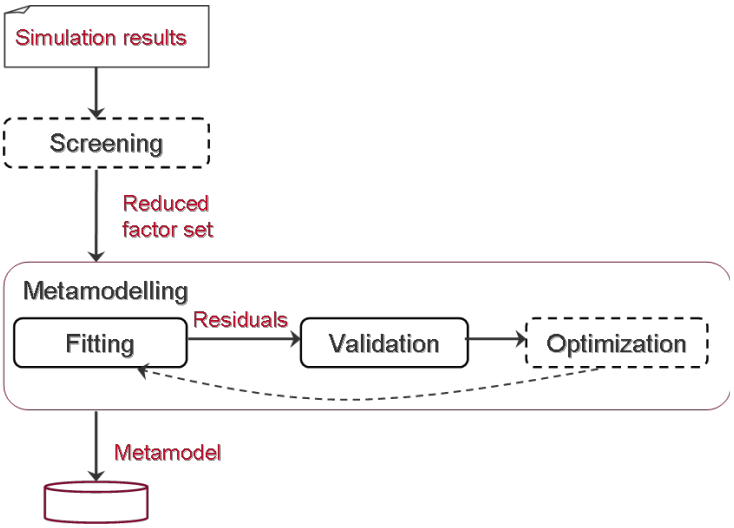


Figure 3.17: Flow to build the metamodel

Section 3.3.1 realizes the factor screening, by analyzing the response variance. It identifies which factors have significant impact and are worth considering in following experiments. Section 3.3.2.1 describes the algorithms used to estimate the coefficient set for the metamodel and Section 3.3.2.2 presents methods to evaluate its fitness. Algorithms to optimize the metamodel are presented in Section 3.3.3. Only after making sure the metamodel is adequate, can it be used to predict the response and for subsequent worst-case analysis.

The approach starts with selecting statistical methods of analysis of variance, metamodelling and residual analysis out of the available set. These are evaluated on deterministic functions and modified when needed, e.g. for the computation of residuals, while the procedures which optimize the metamodels are own extensions of the approach.

Decoupling factor correlations

Factors must be decorrelated before any study of response variance, by decomposing into independent sources of variation. Otherwise, the impact of a factor cannot be differentiated from ones correlated with it, as the orthogonality property of the DoE would not hold (3.2.4.1). There are two types of correlations of interest: when some factors are linear combinations of other factors:

$$F_n = \sum_{i=1}^{n-1} \alpha_i \cdot F_i \tag{3.3}$$

[2]Screening should be involved when the factor set is too large and metamodel optimization when the validation cannot output a good enough model.

F_n is simply removed from the analysis and after regression with respect to the first $n-1$, the metamodel $R(F_1, ..., F_{n-1})$ becomes $R'(F_1, ..., F_n)$, with coefficients modified according to the codependency relation.

Alternatively, in RFDoEs, factors can be generated from already existing factors, to which extra sources of variation are added, for instance as Figure 3.13 shows.

$$F_n = \sum_{i=1}^{n-1} \alpha_i \cdot F_i + \epsilon \tag{3.4}$$

Such weaker correlations can also be produced by passing a correlation matrix when generating the design. In these cases, decoupling is not considered necessary, as the DoE involves sufficient runs to estimate coefficients for weakly correlated factors.

3.3.1 Analysis of variance for factor screening

The analysis of variance (ANOVA) is a formal method which distributes the total response variance into the variance explained by each factor. As recommended by [Mon05, p.160-177], ANOVA is applied here to identify factors statistically significant. It determines which factor effects are nonzero, by estimating preliminary main effects, assumed high only for significant factors. It is therefore performed after the so-called screening experiments on a large set of factors, e.g. after R3FF DoEs, as they were detailed in 3.2.4.1.

The analysis of results procedure evaluates the hypothesis on equality of response values for different combinations of factor levels. This test distributes the sum of squares SS, into the sum of squares for factor F_i, and the left over variance that cannot be explained, i.e. the error term ϵ. Appendix C explains more on this topic. Applying the F-statistic on the sums of squares, ANOVA estimates the probability p for the null hypothesis of equality between factor treatments. Then, important factors can be filtered based on the p values. Table 5.4 in the results shows a reduced form of the ANOVA Table, with the relevant information to make a screening decision. The sums of squares SS and probability p of the tested hypothesis are extracted, for each factor.

Factors are then filtered on the criteria:

- $p(F_i) < 0.05$. A more permissive threshold of 0.1 can also be used.
- $SS(F_i) > \epsilon$. This recommended check makes sure that the error term is smaller than the factor variance, which is also a measure for the factor effect.

There is no hard restriction on the number of factors which must be filtered by screening, but in practical cases of interest here, not more than 10 factors are usually kept after this step. This is based on the sparsity of effects principle and on the consideration that not all factors can have significant impact on a response, relatively to each other.

3.3.2 Fitting the metamodel

The metamodels to fit are multivariate polynomials. On the one hand, the DoE is chosen after defining the metamodel type to fit in this step. On the other hand, it is important to select a metamodelling approach whose performance, to some extent, is independent of the experimental design [MM06]. That is, the regression analysis and metamodel validation steps which are detailed next must be as robust as possible with respect to the input data and the specific form of the polynomial to fit.

3.3.2.1 Regression analysis

It is the first step in estimating the regression metamodel, after simulating a corresponding experiment. The inputs are the simulation results and the associated DoE matrix. The main output is the set of coefficients of the metamodel which was planned to approximate the response. A multiple linear regression algorithm is applied for polynomial metamodels. The following details the general regression method [50], formulated so that it optimally analyzes the experiment results. The matrix form of equation (3.2) for a 2^{nd} order metamodel is:

$$R = c_0 + c_l^t \cdot F + c_{iq} \cdot F \cdot F^t \tag{3.5}$$

where F is the vector of factors of length n, c_0 is the constant term, c_l is the vector of linear coefficients. t indicates the vector is transposed. c_{iq} is a symmetrical matrix with the interaction and quadratic terms:

$$\begin{matrix} c_1^{(2)} & c_{12}/2 & ... & c_{1n}/2 \\ ... & ... & ... & \\ c_{1n}/2 & c_{(n-1)n}/2 & ... & c_n^{(2)} \end{matrix}$$

The vector c of all unknown coefficients, of length $(n+1) \cdot (n+2)/2$, is formed:

$$c = [c_0;\ c_1; ...c_n;\ c_{12}; ..; c_{1n};\ c_{23}; ...; c_1^{(2)}; ...; c_n^{(2)}]$$

Then, the so-called regressors matrix X is built out of the DoE matrix d. X contains a column for each term of vector c. The values for each column are computed as the multiplication factor of the specific coefficient in c, in the specific run. For the matrix $d = (d_{ij})_{i=1,m;j=1,n}$, X is:

$X = [d_{i1}\ d_{i2}\ ...\ d_{in}\ d_{i1} \cdot d_{i2}\ ...\ d_{i(n-1)} \cdot d_{in}\ d_{i1}^2\ ...\ d_{in}^2]$, whose size is $m \times (n+1) \cdot (n+2)/2$.

Applied to the vector of results r of length m, the metamodel approximation becomes:

$$X \cdot c \approx r \tag{3.6}$$

This is an overdetermined linear equation system with the unknown c. Then c is:

$$c = (X^T \cdot X)^{-1} \cdot X^T \cdot r \tag{3.7}$$

When the design is orthogonal, d has pairwise orthogonal columns and $d' \cdot d$ is a multiple of the identity matrix. Then, the solution to the equation is simple, since $(X^T \cdot X)$ is also a multiple of the identity matrix. The general procedure applies the least square fit to minimize the sum of squares of the residuals, i.e. the differences between the results r and the estimated values. That is, c is estimated by solving:

$$\|residuals\| = \|r - X \cdot c\| = min. \tag{3.8}$$

The regression technique can vary e.g. weigh differently different points of the factor space, such as robust fit does, by putting less weight on outliers. More details on regression alternatives and associated statistical measures are available with [52].

3.3.2.2 Validation of the metamodel

In order to conclude that the assumptions initially made on the response are acceptable, the residuals, i.e. the errors of the response predictions, are analyzed. That is why this step is also referred to as residual analysis. It is recommended to invest just a part of the results for regression and leave the rest for metamodel validation.

The inputs of this step are the estimates for the metamodel coefficients, the DoE matrix and the corresponding simulation results, against which the metamodel must be validated. The output is at least a true/false result whether the metamodel is fit. The user interaction can be useful, for visual checks, as seen next. The criteria to evaluate the residuals must be clearly defined.

Sample points for residual analysis

The metamodel validation is first performed with respect to the samples used to build it. Still, validation against a new set of samples is more reliable. This set is either a subset of the initial results, which are used only now, or a new set of results dedicated to the validation step. The new sample points can be placed in regions of special interest e.g. random corners. Alternatively, they can be uniformly distributed or points around the metamodel extremes. The runs initially used for factor screening should also checked by the residual analysis. Since the screened factors do not have a significant impact, the metamodel should be a valid approximation, as long as a higher threshold for error acceptance is used.

It must be checked that the fitting step is, as far as possible, independent of the samples used to do it. The "Leave-K-Out-Method" explained in the next section can decide on this robustness, by varying the sample choices used for fitting and validation. The closer the validation points are to the fitting ones, the more likely it is that the residuals are smaller. When they are equally far to the set used for fitting, a rather small residual variance is expected. More outliers with respect to the metamodel predictions determine an increased variance and a non-zero mean.

Computing the residuals

A commonly used residual metric is the average relative error:

$$\epsilon = 1/m \cdot \sum_{i=1}^{m} |r(i) - R(i)| \,/\, |r(i)| \tag{3.9}$$

where m is the number of runs, and r, respectively R, are the simulated and predicted response for run i. It is also common to norm the residuals to the average response. However, residuals are normed here to the maximum response variation, in order to remove the dependency on the order of magnitude of the response, which is application specific, and use the same decision thresholds. This provides more reusability and consistency to the subsequent evaluation metrics. Therefore, the normed residuals used for validation are computed as:

$$res = (r - R)/(max_{i=1,m} r(i) - min_{i=1,m} r(i)) \tag{3.10}$$

The minimum requirements relate to:

- mean, maximum values

- statistical distribution
- correlations to the response and to the factor values

There are many other ways to compute residuals[3]. Sources in the literature like [SPKA01, KR03, GCLD10] provide more details on alternatives. For the present case, the normed residuals as introduced in equation 3.10 are considered sufficient.

Decision threshold for residuals

The threshold to use for concluding about residuals depends on the experimental phase. 1% has been found a good threshold for the mean of absolute residuals, while 1% to 5% is chosen for the absolute maximum. Values higher than that are considered not reliable, because a prediction error bigger than 5% of the total response variability is unacceptable. The mean and maximum absolute values for the residuals are also used to apply corrections to worst-case predictions.

Figure 3.18 plots a quadratic metamodel on 2 factors, i.e. the surface, and two sets of samples. The green response data is fitted well by the metamodel, while the red data is not. The sets of samples are generated by adding to the response surface random departures. The evaluation criteria under test is given by the residuals. Clearly, the decision is much improved when residuals are automatically computed and compared to thresholds, than in the case of a simple visual inspection.

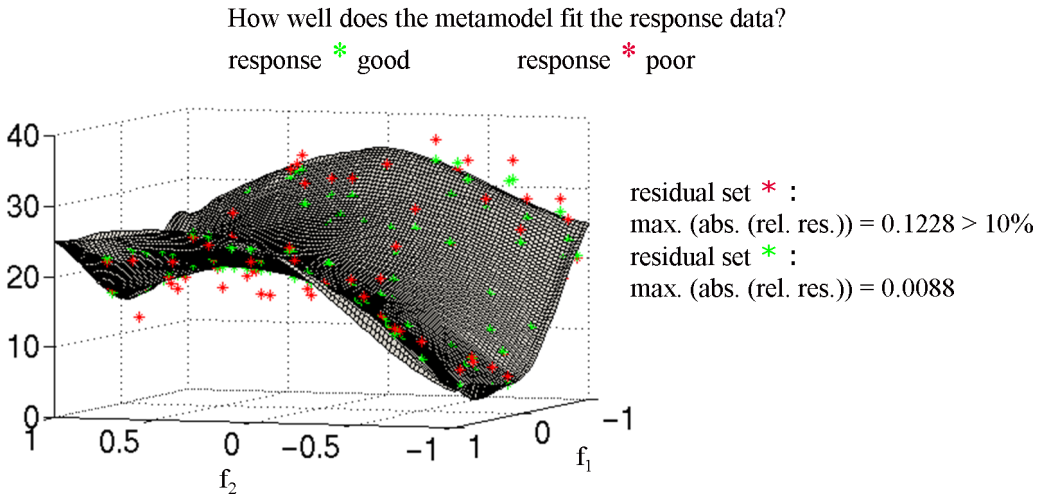


Figure 3.18: Example plots of response data and of the fitted metamodel

Distribution of residuals

Real-life experiments assume the experimental error is normally distributed with mean zero and can decide on the metamodel fitness based on that. This is not proven to always stand in simulated cases, as it is application-dependent [SPKA01]. Still, in our case, the effects which determine departures of the response from the metamodel, e.g. round-off errors, can be approximated with

[3] e.g. normalized root mean squared error, normalized maximum absolute error

white noise and evaluated using the distribution criteria. A normal probability plot can assess how close the residuals are to a normal distribution with mean zero. Such a plot scales the y-axis according to distances between the quantiles of a normal distribution. Residuals of inadequate models are nonlinearly distributed or have nonzero means, which is the center value on the x-axis.

Figure 3.19 shows a comparison of adequate versus inadequate residual sets. The sets are generated as random samples which fulfill the decision criteria good enough (blue stars) or not (red circles). The validity of the null hypothesis H_0 stating that the data does not come from a normal distribution results out of the test, which enables automating the decision process. The results chapter shows how the distribution assumptions apply on large samples of data.

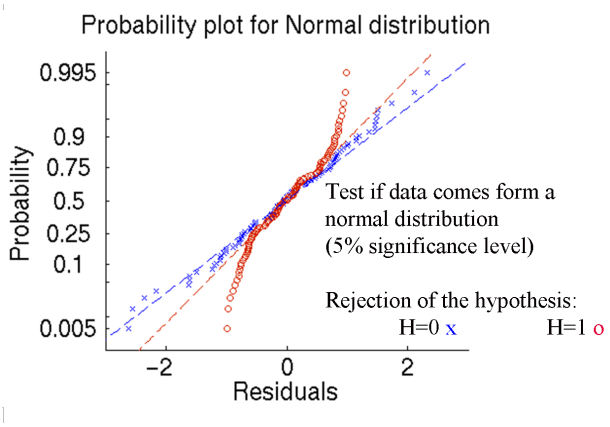


Figure 3.19: Comparison of normal probability plots

Correlations residuals-responses, residuals-factors

The distribution of the residuals with the response values is also important, mainly to make sure there are no visible patterns, therefore no systematic errors of prediction. Scatter plots can also easily detect outliers, in order to judge on the robustness of the regression. Similar analyses can be performed on residuals versus the factor values. The degree of correlation to the response values can be quantified as coefficients. They should be used at least to compare between sets of residuals or to optimize when iteratively fitting metamodels as illustrated in Section 3.3.3. The 0.15 threshold is used to decide between correlated and uncorrelated sets of values. This is found the highest value which still allows correct regression of results on deterministic responses.

Example

The methods presented above can be tested and visualized on deterministic functions. A 10-factor 2^{nd} order multivariate polynomial, with a superimposed error term ϵ, which is distributed randomly $N(\mu = 0, \sigma = 0.1)$[4], is computed to mimic simulation results.

$$R = c_0 + c_l^t \cdot F + c_{iq} \cdot F \cdot F^t + \epsilon \tag{3.11}$$

[4]Normally distributed with mean μ and standard deviation σ

where t means the transposed form, c_0=0; c_l=[1.0 2.0 ... 10.0]; the matrix c_{iq} is:

0.1	0.12/2	...	0.2/2
...	...	...	
0.2/2	0.19/2	...	0.1

The design used for regression is a CCD DoE, which has 149 runs, as Figure 3.15 points out. An LHS DoE of 50 runs is used for validation[5]. Regression is performed on the results of the design, then the residual analysis is applied on both for comparison. Figure 3.20a represents the residuals against the response, stars for the residuals with respect to the initial CCD DoE and circles for the residuals with respect to the validation LHS DoE. The distribution of both sets, represented on a normal probability plot is shown in Figure 3.20b.

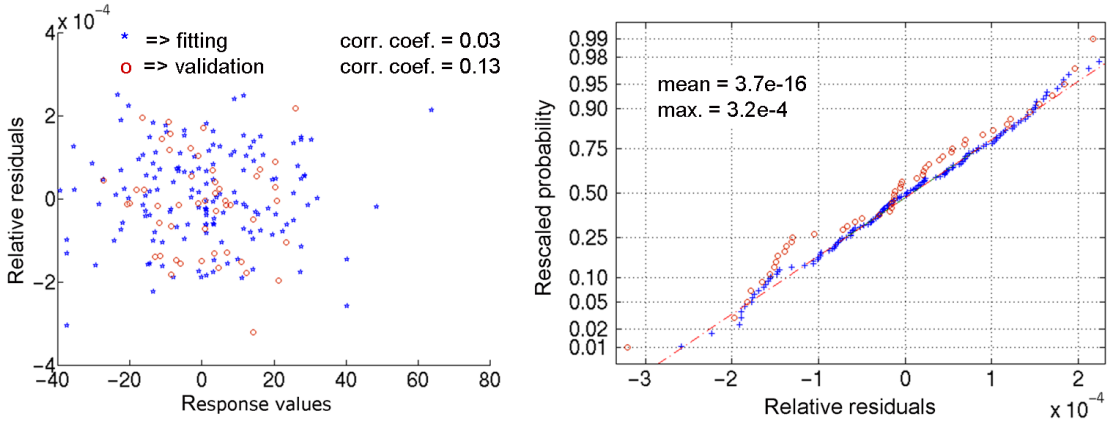


Figure 3.20: Validation results for the example
a. Scatter plots of residuals b. Normal probability plot of residuals

The estimated coefficients have a maximum relative error as compared to the true (initial) set of 0.07, which is considered acceptable. The correlation coefficients residuals-responses and the mean and maximum absolute normed residuals are visible on the plots.

Confidence bounds

The analysis of results can also extract (1-α) confidence intervals for each coefficient [52]. The smaller the α value is, the wider the confidence bounds are. The bounds result into confidence intervals for the response predictions and can be used to improve the worst-case analysis (Section 3.4.3).

Implementation algorithms

The fitting and the validation steps are implemented in the experiment controller by the dedicated functions:
`[coef,residuals_ok] = do_regression(doe_matrix,r,type)` and

[5]The minimum number of runs is chosen based on the minimum pairwise distance between points of the design, i.e. a distance of 0.5. More runs can be used for validation when feasible, but 30% of the regression runs is considered sufficient.

`residuals_ok = validate(doe_matrix,r,coef).`
A description in a pseudocode form is attached in the Appendix A.2. The optimization methods presented next implement loops over these basic functions.

3.3.3 Optimization of the metamodel

The metamodel optimization problem can be defined as finding the optimal metamodel R^*, which:

$$R^* = R \; so \; that \; \epsilon(R_{c,\theta}(d), r(d))_{c\in C,\ \theta\in\Theta} = min. \tag{3.12}$$

with the minimum requirement:

$$\epsilon(R_{c,\theta}(d), r(d)) \leq \tau \tag{3.13}$$

where $R_{c,\theta}$ is the parameterization θ of R and c is the set of R coefficients. ϵ is the error function which represents the residuals and τ is its target value. d is the matrix of points used for optimization and $r(d)$ is the corresponding set of response values. The metamodel optimization problem becomes a selection of the best metamodel parameterization, with respect to θ[6] and with respect to the coefficient set c. The selection of the error function ϵ and of the corresponding threshold τ is as important as the initial metamodel choice, represented by θ and c.

3.3.3.1 Finding the lowest fitting polynomial order

The first optimization step is to iteratively add one order to the metamodel and to regress, until the residual analysis is satisfactory. With respect to definition 3.12, this means optimizing with respect to θ. Figure 3.21 shows this idea.

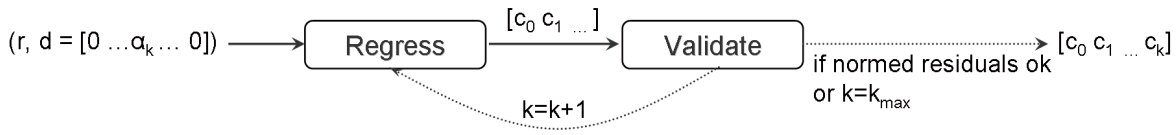


Figure 3.21: Search for the lowest order of an adequate metamodel

The algorithm starts from a 1^{st} order metamodel and iteratively estimates the next polynomial order for each factor, provided enough simulation points are available. Alternatively, including simulation in the loop is possible, but such sequential experimentation methods are treated in Section 4.1. These additional points are axial runs of the respective factor, at least one for each additional order. The regression assumption is still that interactions are of low orders, as in the case of a CCD DoE. Residual analysis is performed and the decision threshold is the same as before. In order to compare factor effects of different orders and decide on their significance, the effects estimates must first be rescaled, then compared.

Figure 3.22 shows how polynomials with increasing orders with respect to one factor can approach the sample points better, even when random terms are superimposed on the polynomial used as response.

[6] Best parameterization e.g. finding the lowest order of fitting polynomial or best matrix of factor effects.

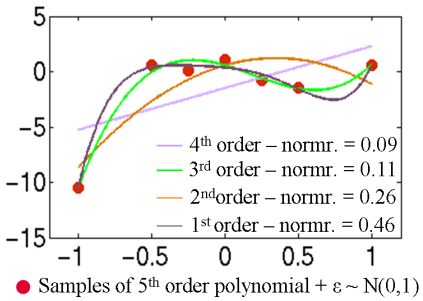


Figure 3.22: Estimation of high-order effects

The metamodel can also be optimized with respect to interaction effects. This would add 2-factor, 3-factor, etc. interactions to the parametrization, regress and measure the improvement. This could be applied when needed, based on the sparsity of effects, which implies that effects become smaller as their orders become higher.

3.3.3.2 The leave-k-out method

This method performs an optimization with respect to the coefficient set c. It applies when only a part of the runs are invested to fit the metamodel, and the rest are used strictly for validation. The Leave-k-out method implements this in a loop, which permutes the validation samples, within the complete set of samples. The principle is visible in Figure 3.23.

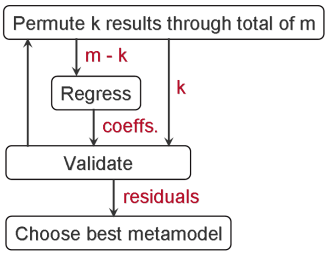


Figure 3.23: The leave-k-out metamodel optimization flow

The loop predicts with respect to $(m - k)$ and validates with respect to the k left points, where m is the total number of samples. The best fit is chosen as the one with the best final residual analysis. To best meet the requirements explained in 3.3.2.2, the mean distance between the samples used to evaluate the residuals and the ones for the fitting must be approximately equal from one iteration to the next.

3.4 Concluding the experiment

This chapter explains how to conclude on the overall experimental results. The first part describes how to analyze the properly fitted metamodel, output of the previous steps. Sensitivity analysis is addressed by translating the metamodel into factor effects, to be compared and interpreted as detailed in Section 3.4.1. The metamodel can be used to make predictions on the response (Section 3.4.2) and to go on with the worst-case analysis, explained in Section 3.4.3. Such an analysis can follow both fixed-level and RFDoEs.

Section 3.4.4 presents an analysis procedure which can be applied only on RFDoEs, for response characterization in terms of factor effects and the worst-case. Correlations to factors estimate their effects. Fitting the response statistical distribution enables predictions on new response values and provides worst-case estimates.

Graphical analysis plays a big role in the complete flow. Figure 3.24 shows the steps followed by these steps of response characterization.

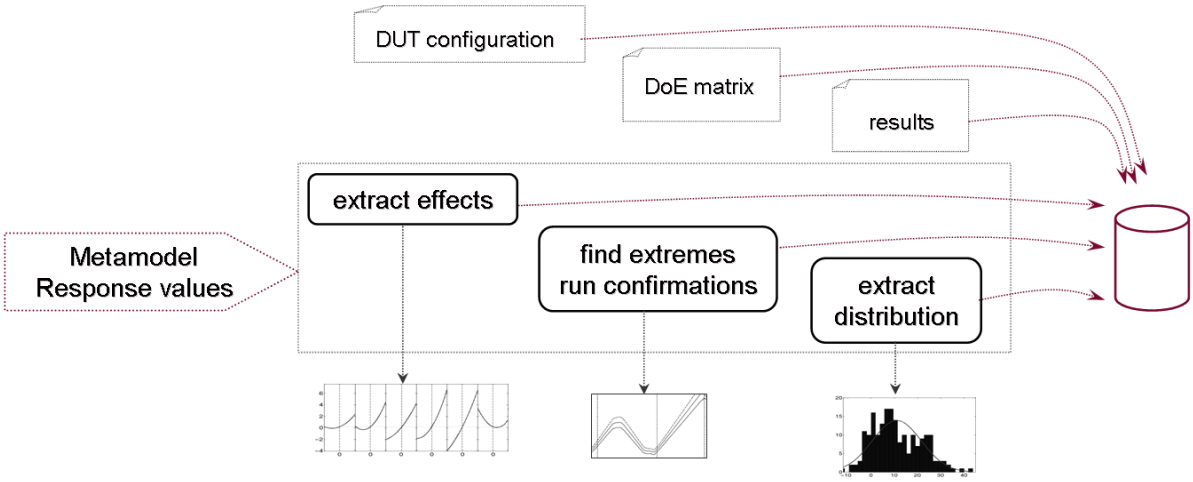


Figure 3.24: Response characterization flow

The approach must extend the available algorithms of metamodel analysis e.g. for decision on which factors are important or to find the metamodel extreme. The analysis of results both for custom DoEs and for RFDoEs are own contributions of the approach.

3.4.1 Factor effects

The sensitivity analysis problem is addressed when the factor effects are quantified in an unified and consistent manner. Then, they can be compared, visualized and interpreted, for a better understanding of the system. In the case of polynomial regression metamodels, factor effects can be directly mapped to the coefficients, as introduced in Section 3.2.2.

Individual factor effects Are the effects which characterize that factor only, i.e. which do not change their value or sign, no matter the levels of the other factors. Linear, quadratic and higher-order polynomial coefficients estimate these effects. To graphically analyze such effects, a representation of the metamodel on each factor, at fixed-levels of other factors is sufficient. Figure 3.25 shows such a representation for a metamodel with the linear coefficients: c_l = [0 1 2 3 4 5 -1 -2 -3 -4 -5] and the quadratic ones: c_q = [1.0 2.0 1.0 2.0 1.0 2.0 1.0 2.0 1.0 2.0].

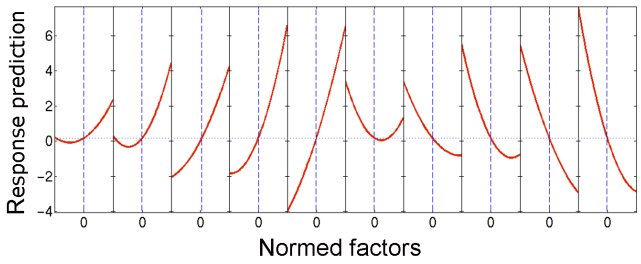


Figure 3.25: Individual factor effects

Interaction factor effects Are the effects which change value or sign when other factors change levels. Such effects are hard to detect with trial-and-error methods, because one must concomitantly vary several factors, in order to track potential interactions. The ability to detect the existence of such effects is one of the biggest benefits of the DoE methodology, with the associated experiments and metamodels. 2-factor interactions are most common and should be visualized for a correct interpretation. Interaction plots show the dependency of the fitted metamodel on selected factors, at various levels for the other factors. For example, for the metamodel plotted in Figure 3.25, the interaction effect between factors 1 and 2 is visible in Figure 3.26.

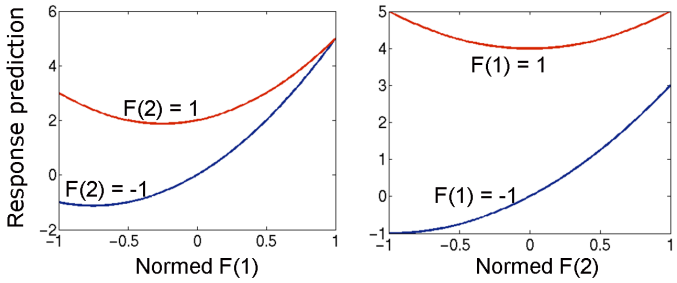


Figure 3.26: Interaction factor effects

MATLAB enables an interactive plotting of regression models of up to the 2^{nd} order: the predicted response values are plotted on each factor, at fixed levels of the others. The interactive nature allows updates of the dependencies on all factors and of the confidence bounds, when modifying the level of one factor. Figure 3.27 shows an example of such a plot, after regressing the results of a DoE on 10 factors, applied on the response:

$$r = 0.1 \cdot \sum_{k=1}^{10} k \cdot F(k) + 0.075 \cdot \sum_{k=1}^{10} k \cdot (-1)^k \cdot F(k)^2 + \sum_{i=1}^{9} \sum_{j=i+1}^{10} (-1)^i \cdot F(i) \cdot F(j) + \epsilon \qquad (3.14)$$

where ϵ is normally distributed N(μ=0, σ=0.05). The DoE is composed from a 149-run CCD and a 50-run LHS, with normally distributed factors.

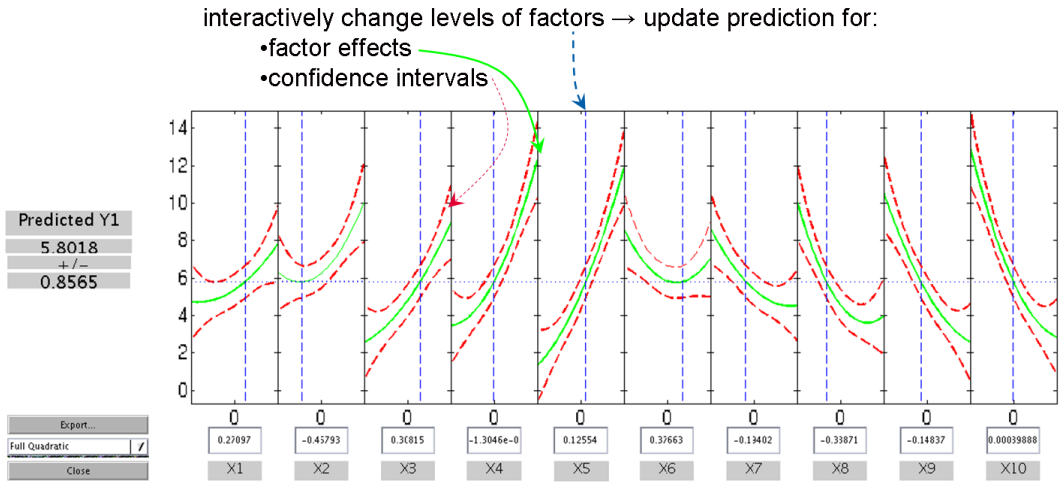


Figure 3.27: Interactive prediction plot

For each coefficient, $(1-\alpha)$ statistical confidence bounds are estimated. These reflect into bounds for the predicted response, which can be visualized on the interactive plots. Since the regression data is normally distributed, there is naturally more confidence in the center point of the factor space and it decreases as approaching the corners[7].

In order to judge on significant factors, effects must be compared. Effects can be directly compared only when they are of the same type i.e. linear coefficients, 2-factor interactions etc. A linear coefficient is considered significant here when a variation in the factor alone with 10% of its range causes a response variation of at least 1% of its total relative variability. That is, if the following is fulfilled:

$$abs(c_l) \;>\; (max_f R(f) - min_f R(f))/(20 \cdot abs(R(0))) \tag{3.15}$$

For the interaction and quadratic coefficients, applying the same condition of at least 1% of response variation on the equation of the metamodel yields the condition:

$$abs(c_{ij}) \;>\; (max_f R(f) - min_f R(f))/(4 \cdot abs(R(0))) \tag{3.16}$$

Factor effects in the custom DoE

A special analysis must be performed on the custom DoE introduced in Section 3.2.4.1. With respect to the main factor, regression can extract higher-order effects. Alternatively, the simulation results can be interpolated over the main factor levels for a piece-wise linear $response(mainfactor)$.

This DoE actually focuses on the interaction effects between the main factor and the set of other factors. The plots of the response, over the main factor levels, for different factor sets, can

[7] With factors otherwise distributed, for instance corner-peaked as shown in Section 3.2.3, the confidence is higher in the area of higher sample density, i.e. near the corners.

highlight such high-order interactions. It indicates interaction by lack of parallelism between plots of the response for different factor sets. An important interaction would determine a response extreme i.e. the worst-case, at a level of the main factor which depends on the factor set. Another sign of interaction appears when the response variability on the main factor is different for different sets. Figure 3.28 shows such an interaction plot, for a DoE which uses 10 levels for the main factor and 2 for each factor of the factor set.

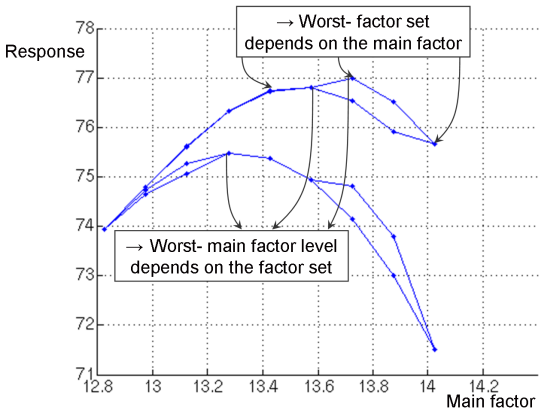


Figure 3.28: Effects in a custom DoE

The main effects for the main factor and for the factor set, as well as the interaction between them, can be estimated. The response's mean and standard deviation, for different levels of the main factor and for different factor sets, are used. The change in response over main factor levels, relative to the distance between different sets, corresponds to the ratio of main effects (main factor/factor set). The contrast between deviations within sets at different main factor levels measures the interaction between the main factor and the factor set. These estimations of factor effects are summarized in Table 3.3. Table 5.1 shows results of such an analysis.

Table 3.3: Effects in a custom DoE

Effect estimates	
Main factor effect	mean(std. within sets)
Factor set effect	std(mean within sets)
Interaction	std(std. across main factor levels)

3.4.2 Prediction of the response

When an adequate response regression metamodel is built, a large benefit is the ability to replace expensive simulation runs by fast evaluations of the metamodel in sample points of interest. After a proper validation, provided that each factor is in the $[-1, +1]$ range, the following approximation holds:

$$r(F) \approx c_0 + \sum_{k=1}^{2} \sum_{i=1}^{n} c_i^{(k)} \cdot F(i)^k + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot F(i) \cdot F(j) \tag{3.17}$$

The regression analysis provided confidence bounds for the coefficients. They can be easily extended to provide bounds for the response prediction, which is also visible in Figure 3.27.

Given a factor set, these response bounds are computed as:

$$r_{min}(F) \approx c_{0min} + \sum_{k=1}^{2}\sum_{i=1}^{n} min\{sgn\{F(i)^k\}\cdot c_i^{(k)}\}\cdot F(i)^k + \sum_{i=1}^{n}\sum_{j=i}^{n} min\{sgn\{F(i)\cdot F(j)\}\cdot c_{ij}\}\cdot F(i)\cdot F(j) \tag{3.18}$$

A correction for these bounds can be the residual in the center ($F = 0$), as it has a high significance, i.e. the prediction offset for the nominal simulation. Alternatively, the mean or maximum residual can be used for correction. The relative distances between the point of prediction and the ones used for metamodel fitting can also be accounted for, e.g. by using radial basis functions [MM06].

3.4.3 Worst-Case response prediction

The worst-case response is most of the times defined as one of the two global extremes of the response over the factor space. Once the metamodel is available, it becomes subject to deterministic optimization[8] and its extreme is the estimated worst-case. The problem can be simplified by considering only the minimum of the metamodel R. The solution to finding the maximum is similar, but performed on $-R$.

Figure 3.29 illustrates the ideas discussed in the following.

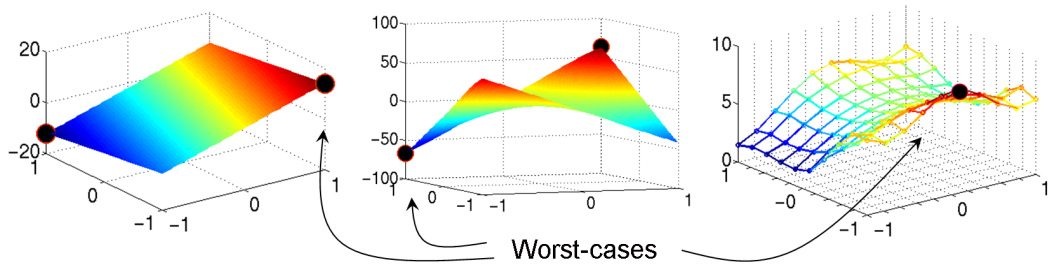


Figure 3.29: Worst-cases in different metamodels
a. linear b. with interactions c. 2^{nd} order model

In the case of a linear metamodel, the solution is simple, since the worst-case of the hyperplane represented by the metamodel is a corner. It can be easily computed by setting the factors to either 1 or -1 depending on their linear coefficient. When only linear and 2-factor interaction terms are present, the worst-case is also a corner. As noted in the middle plot in Figure 3.29, the corners can be relatively close to each other because of interactions.

In cases where all corners are too many to be evaluated fast on the metamodel equation or quadratic terms are present as well, a constrained optimization method is applied to search for the minimum of the metamodel. To look for the minimum of a function of several variables starting at an initial estimate, a gradient-based algorithm is commonly used, when the function is continuous and has continuous first derivative. This requires an iterative procedure, and determines the direction of search at each major iteration.

[8] Finding the extreme of an objective function is commonly referred to as optimization.

Nonlinear Programming addresses the optimization problem for an objective function nonlinear on the factors. Quadratic Programming is a particular case of interest, of a quadratic objective function that is linearly constrained. The procedure finds an initial feasible solution, by solving a linear programming problem. Then, it generates an iterative sequence of feasible points that converge to the solution. Sources like [50] make such algorithms available and provide more details about their implementation.

The algorithm can present convergence problems e.g. because of interaction effects. A solution which addresses this issue and can even reduce the optimization time is, when feasible, to evaluate the metamodel on a hypergrid. The right plot in Figure 3.29 shows how this first narrows down the space, then the worst-case is searched within the hyper-cubic subregion which contains the minimum found so far. A problem can be the exponential increase of the number of points to be evaluated with the number of factors and a power increase with the number of factor levels per grid: $N = levels\_per\_grid^{number\_factors}$. The same problem can occur when evaluating all corners. The impact on performance is evaluated as negligible, as long as the total number of evaluations is kept smaller than 10^5, i.e. up to all corners for 15 factors can still be handled quite fast. Considering the factor set is already reduced in the step of worst-case search, this restriction is acceptable.

Worst-case corrections For safer response margins, the predicted response extremes can be adjusted with confidence bounds and/or residuals, same as applied to the predicted response values. The residual in the center is most commonly used, when corrections are required.

Confirmation runs Running confirmation simulations in the predicted worst-case is of course of interest. Comparing the result to the predicted value and to the previous simulations offers more confidence. Low residuals, i.e. below a threshold of 1 to 5%, confirm the metamodel is well fitted in the neighbourhood of the worst predicted case.

3.4.4 Concluding after random factor DoEs

This section describes solutions to estimate the factors' effects, as well as the distribution and the worst-case for the response, on results of random factor DoEs (RFDoE).

3.4.4.1 Factor effects

The correlation between the vectors used for the factors in the experiment and the vector of response results can be estimated and used to characterize the response in terms of significant factors. Therefore, correlation coefficients can measure factor effects, and can be used alternatively to the coefficients of the fitted metamodel. This best suits the RFDoEs, where a metamodel is rarely formed, but data with an accurate statistical distribution is collected.

More options for estimating correlation are available:

- The linear correlation, quantified by the Pearson's correlation coefficient, indicates the strength of a linear relationship factors-response.
- Rank correlation coefficients, which do not focus only on the linear nature of the dependency.

- More complex descriptions, e.g. distance correlation.

Details on these type of correlations can be found in sources such as: [52, 58]. The important aspects are that coefficients lie between -1 and +1, and that when an increase in the factor determines a response increase/decrease, then the coefficient is positive/negative. Since main factor effects are considered, by assumption, significant as compared to higher order effects, a special focus on the linear dependency should be put anyway. Therefore, the linear correlation is considered sufficient.

For instance, the polynomial: $R = c_0 + c_l^t \cdot f$, where c_0=0 and c_l=[1.0 -2.0 3.0 -4.0 5.0], is subject to an LHS DoE of 200 tests on the vector of factors F (t=transposed). The correlation coefficients between the sets of factor values f and the results computed for r are: $corr(r, f) = [0.230; -0.381; 0.463; -0.564; 0.701]$. Another example can be found in the results on the window lifter, i.e. in Section 5.2.3.

The values of the coefficients in the example are proportional to the main factor effects and have the same signs. Their signs indicate the direction of the effect, while the values of the coefficients, as related to each other, indicate the magnitude of the main factor effects. Testing the hypothesis of no correlation between the responses and the factors can objectively identify significant factor effects. A probability threshold of 0.05 is used, just as in the ANOVA case. The disadvantage with correlation is that no interaction or quadratic effects can be detected when using the linear correlation type. More complex correlations are possible, however, with available statistical packages, e.g. [50].

3.4.4.2 Response distribution

Correlations of response values to factors can estimate factor effects after an RFDoE, as previously explained. But they are not useful to describe the response standalone, have no predictive nature with respect to the response, and do not estimate the worst-case response. An alternative to regression which still meets these goals is proposed here. Once a set of response values is available, the approach characterizes the response as an independently distributed variable and estimates the response distribution. Its fitted distribution has both predictive power for new response values and can estimate the worst-case as explained in the following.

It is of interest to identify regions where the response data tends to cluster, as well as tails of its distribution. The analysis should be applied on RFDoEs with sufficient data to obtain statistically significant conclusions, provided that factors are distributed as close as possible to their original distribution. The response distribution is estimated, so that the values or intervals with high/low probability of occurrence are identified. By interpolating the empirical probability density function (PDF), the frequency of occurrence for specific values can be estimated. More importantly, the worst-case is then estimated as the highest/lowest value which can occur, with a probability higher than a given threshold.

Fitting the distribution

When the data resembles a well-known distribution, parameters of the specific distribution must be extracted. The functions used to generate the factor distributions, such as the ones presented in Section 3.2.3, can be fitted with respect to their parameters, then used to generate further values.

The simplest cases are the uniform and the normal distribution, where a range, or respectively, a mean and standard deviation, are sufficient. For instance, Figure 3.30 shows a fitted normal distribution.

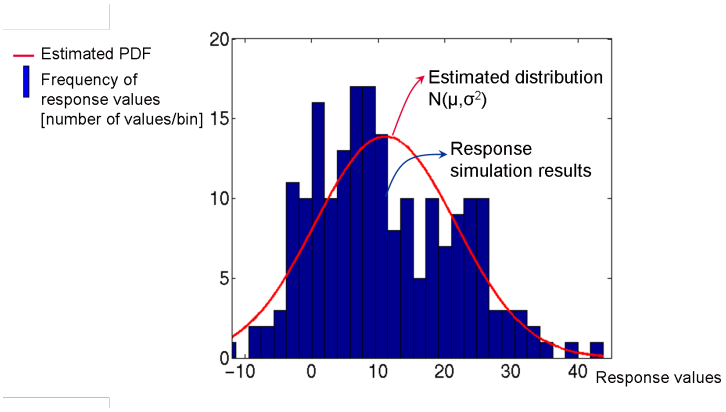


Figure 3.30: Response distribution example

The estimated normal PDF can be quite poor, as observed in the figure. Therefore, more distribution types must be available, including nonparametric ones, as introduced in the following. Moreover, a step of validation is performed, as described in the next subsection. In addition to fitting the custom distributions used for factor generation, such as the ones presented in Figure 3.11, the set of functions contained in the MATLAB statistics toolbox [52] provide support in fitting various tendencies of the response data set. A set of such functions can be found in Appendix B.4. The optimal distribution type is identified and the set of parameters which best fit the response values is estimated.

Nonparametric fitting is an alternative with the benefit that it can model almost any shape of response distribution. Kernel density estimation function [60], also available in the MATLAB statistic toolbox [52], interpolates the empirical PDF and computes the probability density estimate for any sample of the response. Parameters which can be generally used for response characterization are only the mean and variance, but the main advantage is a high accuracy in estimating the probability of occurrence of an arbitrary value. The worst-case is then estimated by extrapolating the fitted PDF, as seen in the final step explained in this section.

Validation

To check the quality of the estimates, the goodness of fit must be evaluated. For this, a residual probability of occurrence (res_prob) is evaluated, i.e. the differences between the estimated PDF and the empirical frequency of occurrence are computed, for intervals in the response range.

$$res\_prob = (estim\_cdf(r_1) - estim\_cdf(r_2)) - (\#(r1 < r < r2))/(\#r) \tag{3.19}$$

$estim\_cdf$ is the estimated cumulative distribution function. $\#(r1 < r < r2)$ is the number of response values between $r1$ and $r2$. $\#r$ is the total number of response values. res_prob is evaluated on equidistant intervals between the response extremes, and its mean and maximum measure the goodness of probabilistic fit. Generally, a 1% threshold to decide on how well a distribution fits on a data set is used. Figure 3.31 shows an example of two distribution types fitted on an arbitrary data set: a parametric distribution (extreme value type), as well as a

nonparametric, better fitting one. The mean and maximum residual probability values are also marked.

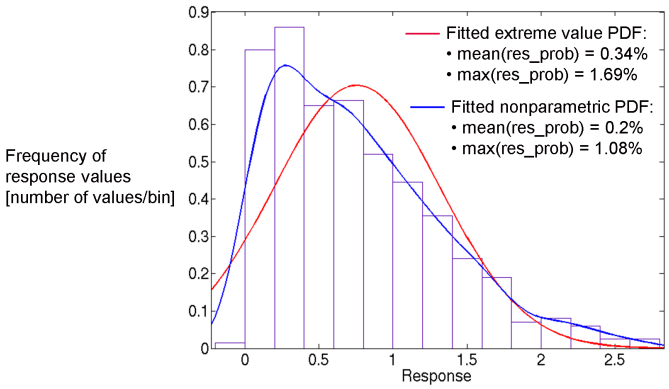


Figure 3.31: Fitted distribution examples

Since the analysis considers the response a random independent variable, it ignores correlations to the distributions of factors, which will always exist, and changing them would most likely change the output of this step. Characterizing the response distribution can also involve mapping it to the statistical properties of factors. This depends, however, on each distribution type, for the factor set and for the response. If this mapping would be established, modifying the factor distributions to identify changes in the response distribution, especially to obtain extreme response values, would be a worst-case search strategy. This is a possible topic for future work, as noted in Section 7.2.

3.4.4.3 Response prediction

The ability to predict response values from factor values is limited when there is no regression model. Still, the estimated response distribution not only characterizes the response, but can also generate further values as a substitute to results of further expensive simulations. The response PDF which has been estimated can predict the probability of each value and it can be used to produce new values with the same distribution.

This response "model" is independent of the factors. The correlations to factors store the sensitivity information. This dependency can be decoupled and several parts of the simulated system can be replaced with independently distributed variables. This can transfer the response characteristics to the next upper-level simulations and speeds them up, because responses can become factors in the next experiment. This steps into the next level of abstraction, replacing the impact of factors with keeping the response distribution accurate.

3.4.4.4 Worst-case prediction

After a response distribution, parametric or not, is well-fitted on the response data, the worst-case is estimated by extrapolating the PDF. More precisely, a specific value estimates the response maximum if the probability to exceed that value is smaller than a specified ϵ. Similarly

for the minimum:

$$max(r) \approx r\_max \Leftrightarrow 1 - estim\_cdf(r\_max) < \epsilon \tag{3.20}$$

$$min(r) \approx r\_min \Leftrightarrow estim\_cdf(r\_min) < \epsilon \tag{3.21}$$

An ϵ value between 10^{-3} and 10^{-6} is used here, depending on the smoothness of the PDF towards its extremes. r_max and r_min can be computed by inversing the estimated CDF or by iteratively increasing r_max or decreasing r_min, and estimating its probability. The *ksdensity* MATLAB function provides this estimate.

For instance, the polynomial $R = c_0 + c_l^t \cdot f$, where c_0=0 and c_l=[1.0 -2.0 3.0 -4.0 5.0], is subject to an LHS DoE of 200 tests on the vector of factors f. The real worst-case is given by: $min(r) = -15$, $for\ f_{r=min} = [-1\ 1\ -1\ 1\ -1]$; $max(r) = 15$, $for\ f_{r=max} = [1\ -1\ 1\ -1\ 1]$. The fitted distribution presents a $max(res\_prob) = 18.3 \cdot 10^{-3}$. The worst-case estimates, for an ϵ value of 10^{-6}, are: $max(r) \approx 15.33$; $min(r) \approx -17.32$. Therefore, the maximum error relative to the maximum response variability is of: 7.7%. The general performance depends on the number of samples, the number of bins used to cluster them and on the smoothness of the PDF. Examples for such worst-case estimations can be found in the results: Sections 5.2.3 and 5.3.4.

3.4.5 Summary of the experiment

Therefore, concluding the experiment mainly focuses on the response characterization, which highly depends on the DoE type. The main steps are:

- Extracting the factor effects from the metamodel, or directly from the results as correlation coefficients; plotting individual and interaction effects, when possible.
- Estimating the response distribution for RFDoEs
- Estimating the worst-case, as an extreme of the metamodel or of the distribution function.
- Running confirmation simulations, when possible, and checking the residuals.

A report folder is created where the controller saves the simulation inputs and outputs, the DoE matrix, the coefficient set of the metamodel, when available. The fitted PDF, the worst-case and estimates for the factor effects, as well as important plots are also stored.

Specification conformance check The worst found case for the response is checked to be compliant with the expected range. The potential reasons for nonconformance are the factors with significant effects.

Prediction model When available, the metamodel coefficients and their bounds are saved for later use or refinement. Given a factor set, the metamodel is simply loaded and used as predictor, by estimating the response in any given point.

Performance analysis A final analysis of the experiment is realized, summarizing the resources and the performance. The resources are given by the number of runs, because the time to plan and analyze the experiment is insignificant as compared to the simulation time, for the systems under study. The final number of runs is influenced mainly by the number of factors and should not be significantly higher than the values provided in Figures 3.15 for fixed level DoEs, and 3.16 for RFDoEs. Table 3.4 describes the performance criteria used to classify the experiment.

The first class of performance metrics were introduced in Section 3.2.5. Figure 3.16 presented the dependency between the distance in the factor space and the number of runs, for RFDoEs. A distance smaller than 0.3, i.e. which corresponds to 500 runs for 10 factors, corresponds to a high coverage. A distance of 0.5 is considered medium, because it is the design with the smallest distance, which still outperforms the classical DoEs as number of runs. In addition, it allows correct regression of results on tested deterministic responses. The thresholds used for the correlation between independent factors are chosen on the same criteria: regression can be correctly performed on data with a correlation of up to 0.1, i.e. such a correlation is still acceptable as residuals, unlike in the case of higher values.

The second class of metrics relate to the metamodel fitness, given by residuals, and are discussed in Section 3.3.2.2. The thresholds for maximum and mean values, for the distribution of residuals and for the correlation residuals-response were tested on deterministic models in the examples in Section 3.3. The third criteria, relevant for RFDoEs, determines how accurate the statistical properties of factors were represented. The thresholds for the accuracy of PDF estimation is introduced and exemplified in Section 3.4.4.2. Finally, with respect to the worst-case estimate, the experiment has a high performance when the distance between the initially simulated worst-case and the predicted worst-case is large enough. A 10% distance is still high, because it means one order of magnitude smaller than the maximum response variability. The simulation residual for the worst-case prediction, when available, should be not farther from the prediction than the maximum residual, hence the used threshold.

Table 3.4: Performance

Performance	High	Medium	Low
Experimental design performance			
Distance between sample points (only RFDoEs)	< 0.3	< 0.5	> 0.5
Correlation between independent factors	< 0.05	< 0.1	> 0.1
Metamodel fitness (only metamodel-based DoEs)			
Maximum residual	$< 1\%$	$< 5\%$	$> 5\%$
Correlation residuals-response	< 0.15		> 0.15
Distribution of residuals is normal	yes		no
Distribution fitness (only RFDoEs)			
Factor correlations are considered	yes	no	
Statistical factor properties are considered	yes		no
Maximum residual of PDF estimate	$< 1\%$	$< 5\%$	$> 5\%$
Worst-case analysis			
δ relative improvement of the worst-case estimate, from the initially simulated worst-case	$> 10\%$	$> 1\%$	$< 1\%$
Residual of worst-case estimate (only metamodel-based DoEs)	$< 1\%$	$< 5\%$	$> 5\%$

4 Extensions of the DoE flow for more complex responses

All the following methods are contributions of the present work which extend the previously introduced methods to handle more complex responses. The goal is either to improve the response metamodel, to optimize the worst-case estimate or to handle time-variant responses. Section 4.1 describes how to embed the sequence of experiment steps into an automated sequential flow. An improvement of the worst-case analysis is presented in Section 4.2, using as starting point the available algorithms of gradient-based optimization. Section 4.3 extends the experiment to a transient response analysis i.e. applies the steps of experimental analysis to each time sample of a response signal. It requires the same number of runs as the basic experiments which were presented. This extension is of interest to observe how the effects of important factors and the worst-case conditions change in time. It provides predictions on the worst-case behaviour of the response over the complete duration of the test. These build safe margins for the signals of interest and can be confirmed by simulation.

4.1 Sequential experimentation

This type of experiment refers to sequential design strategies, which are necessary to resample the factor space depending on the results of the previous experiment. Even in flows which go as initially expected, when the number of factors is big enough, more steps of experimentation need to be followed[1]. Classical approaches also recommend investing no more than 25% of the resources in the first experiment [Mon05, p.1 - 22]. The experimental sequence removes the user interaction, which, although useful, is still optional. The decision-making steps can be automated, provided that thresholds and default settings are first set.

Figure 4.1 shows the necessary steps. Phases common to any experiment are visible, but loops connect them. The loop within the Metamodelling step refer to the sequential strategies which iterate through the results and optimize the metamodel estimates, e.g. the Leave-k-out method, discussed in Section 3.3.3. The external loops indicate sequential design, i.e. iterative sampling or revising the initial experiment settings, e.g. by a reduction of the initial factor set. They should only be applied when the existent data cannot fit any of the available metamodels. Steps which require simulations are marked in gray boxes.

[1] pilot runs, runs for screening, for fitting, for validation, worst-case confirmation runs

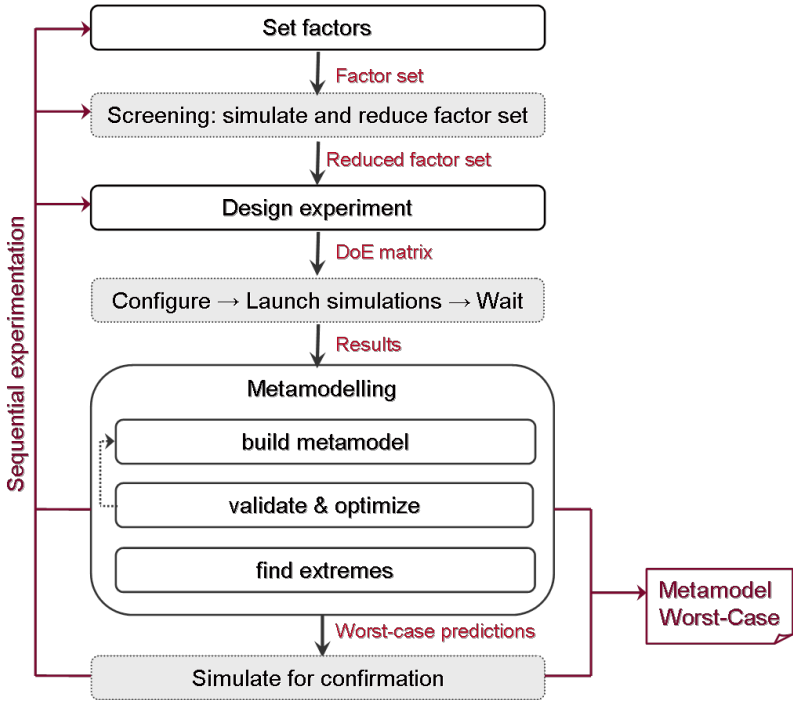


Figure 4.1: Sequential experiment flow

As also noted in [GCLD10], an adaptive sampling must define a sampling function *doe* that constructs a hierarchy $doe(0) \subset doe(1) ... \subset doe(n)$ of nested subsets from the total combinations of factor levels. $doe(0)$ is the initial experimental design and is constructed using one of the available algorithms. An important requirement of *doe* is to minimize the number of sample points added from one iteration to the next, yet maximize the information gain of each successive sampling level. This process is referred to as sequential design, but is also known in literature as adaptive sampling, active learning, reflective exploration or optimal experimental design [WS07]. The advantage of such approaches is that the number of required data points does not need to be specified from start, avoiding potential over/undersampling. An important consequence is that the task of finding a proper response metamodel R becomes a dynamic problem, both with respect to the parameterization and to the set of optimal values of metamodel coefficients.

The main indicators that a metamodel is not appropriate are the residuals and they are used as optimization criteria. To embed the relevant decisions in an automated sequence, the need for user interaction must be absent, e.g. by visual inspection. The metrics associated with residuals, together with predefined thresholds, allow these decisions. Section 3.4.5 justified these thresholds chosen as the boundaries between good and poor performance with respect to the metamodel's fitness (Table 3.4). Therefore, the decision criteria is:

- the absolute maximum of relative residuals ($< 5\%$)
- true result of hypothesis test on normal distribution of residuals, with mean zero
- correlation to the response values ($< 0.15\%$)

When the metamodel is unfit, the steps detailed in the next paragraphs are necessary: revise the assumptions, i.e. isolate the reasons for metamodel unfitness; design the new experiment, be it by a resampling on the same factor set or on a modified factor set; reuse the previous runs for the new metamodelling.

Revise the assumptions

When residuals are not compliant, there is either not enough data, or some assumptions about the metamodel are false. Common sources for these problems are:

- The experiment was not properly designed e.g. not enough runs, or factors are strongly correlated and regression does not decorrelate them.
- Some higher order effects were not accounted for in the metamodel.
- Some interactions were not accounted for in the metamodel.
- The metamodel included too many factors.

The following corrections can be applied:

- Modify the metamodel to fit (a) i.e. add higher order effects or other types of effects. The experiment must then be augmented with the extra runs to estimate the new effects.
- Run more simulations, but keep the metamodel assumptions and the factor set i.e. optimize the design (b).
- Modify the initial set of factors or the factor ranges (c).

The pseudo-code inserted in Appendix A.3 shows how the sequential experimentation is implemented, making use of concepts and algorithms introduced in previous chapters. The sequential follow-ups (a, b, c) are referenced in this implementation.

Choose the next experimental design

This choice depends on the decision previously made about the assumptions to revise. The next experimental design is defined by the resampling function *doe* (also included in the Appendix A.3). When the metamodel parametrization is revised (a), a sequence of DoEs is performed, and the corresponding metamodels are fitted: 1^{st} order model without interactions (R3FF DoE); 1^{st} order model with interactions (R5FF DoE); 2^{nd} order model with interactions (CCD DoE). When the metamodel must be further refined, the next effects to add are higher orders of individual factors, estimated after running more axial points for the specific factor. When specific interactions must be estimated, a higher resolution fractional factorial is performed. When the coefficient set is kept and the metamodel is optimized (b), the next experimental design is an RFDoE, when statistical properties of factors are provided or a space filling, LHS DoE, when they are not. The next decision is to revise the screening step, i.e. to eliminate factors (c). This removes from the factor set the ones with the smallest main effects, which are likely to act as big sources of noise. When the set has been reduced and a proper metamodel has been fitted, in case more runs can still be afforded, the factor set is successively built back. The last resort is to reduce the factor ranges. This is not included in the automation since it implies applying modifications to the initial requirements.

Reuse the runs

Before redesigning the experiment and even more, before performing any new run, it is important to check how to reuse previous runs. If a factor is removed after screening, it is assumed insignificant. Therefore, the previous runs can be used in the experiment on the remaining factor set, considering the screened factor just a "don't care" or a small source of noise. In this sense, as Montgomery notes [Mon05], and as is detailed in Section 3.2.4.1, fractional factorials have the projection property, i.e. they can be projected into stronger (larger) DoEs in the subset of significant factors. This has an application in sequential experimentation: it is possible to combine the runs of two or more fractional factorials to assemble sequentially a DoE to estimate factor effects and interactions of interest. To take advantage of this property, one must reestimate the metamodel, but this time with respect to the subset of factors for which important main and interactions effects were found. The direct benefit is that more complex interaction effects for this subset of factors can be estimated, without loss of accuracy in the results and without investing new runs.

In a reverse situation, when a new factor must be taken into accounted, the previous factor set is a subset of the new one. Therefore, the previous experiments are subsets of designed experiments for the new factor set. Old runs are used in the new setup, with the new factor simply set to 0. Although they do not help in estimating effects with respect to the new factor, any previous run can be reused, at least for validation of new estimates.

4.2 Optimization of the worst-case by gradient-based search

To increase the confidence and correctness on the worst-case analysis, further simulation runs can be invested to search in the neighborhood of the previously estimated response extreme. A gradient-based search is implemented, similar to what has been introduced in Section 2.3.3. The self-correcting search method takes as starting point the factor set of the worst-case found so far. An iterative sequence of simulation sets is performed. The search undertakes the role of the simulation controller, initiating the launching of each simulation set. While performing this search, it allows no inputs other than the results of the previous simulation set.

The problem addressed here is similar to the optimization problem presented in Section 3.4.3. The worst-case can be considered for simplification the global minimum of the response over the factor space. The procedure makes similar assumptions on the objective function i.e. continuity, derivability, to induct the next search direction. The direction of search is determined at each major iteration, by performing a metamodelling step for each simulation set. A constrained optimization algorithm is then called on the estimated function.

However, the main difference as compared to the previous problem is that the objective function is not the deterministic metamodel, but an unknown function, i.e. the response of the simulated system. The constraints are stricter, as the size of the search area is reduced. It should be chosen inversely proportional to the accuracy of the metamodel estimated in previous steps. Specific differences which relate to the algorithm options are detailed next.

It is important to provide the algorithm with options which fit the given problem. Although these are usually application dependent, thresholds and ranges which were found effective on the responses under study will be mentioned. One of these options is the starting point of search: it is set to the previously estimated worst-case, i.e. the output of the step explained in Section 3.4.3.

The other is the search step, i.e. the variation applied in the factors, commonly 10^{-3}. The main options, however, relate to the stop criteria. The algorithm returns when one of the following is met:

- relative change in response during the last iteration smaller than a specific threshold. A small change in the response must be expected in reduced regions of search, so this stop criteria is set relatively low e.g. 10^{-6}.
- maximum number of iterations reached (e.g. 10 to 20). Alternatively, the maximum number of runs can be set, which is preferred since the size of each iteration increases with the number of factors e.g. 100 runs.
- violation of factor constraints, i.e. the hypercube of radius one.
- changes in factor predictions smaller than a given threshold e.g. 10^{-6}.

Sources like [51] make such algorithms available and provide more details about their implementation. The MATLAB functions $fmincon$ and $fminsearch$ are used for implementation. A handle to the simulation controller is passed as parameter.

Figure 4.2 shows a test of the algorithm on an approximately polynomial function on 2 factors:

$$R \approx 1 + 10 \cdot f_1 - 5 \cdot f_2 + 5 \cdot f_1 \cdot f_2 - 10 \cdot f_1^2 + 25 \cdot f_2^2$$

The response is replaced with this deterministic function which is the represented surface in the figure, added to an independent error term, which models real departures from the metamodel.

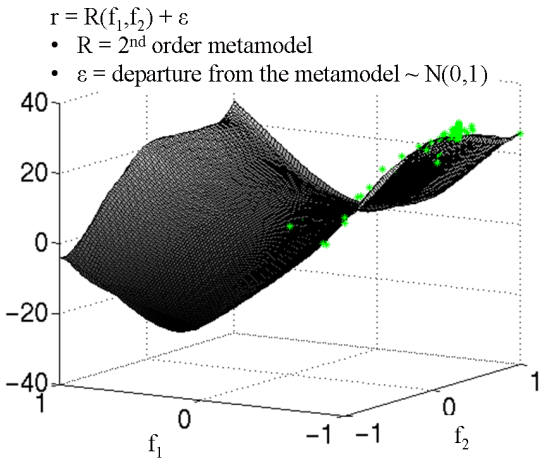


Figure 4.2: Worst-case search example

For simplification, the center is the starting point and a bigger search step is used. The gradient-based search can be observed: after a few iterations around the central area, the search is conducted into the steepest direction, approaching the maximum of the polynomial. A similar search is applied on the real system response in Section 5.3.5. It is compared to a direct search in the full initial factor space. The algorithm is efficient in improving the worst-case results only when applied on a reduced area, i.e. as explained in this section.

4.3 Extension for transient response analysis

Simulation tests how the system under study responds in time, under a predefined stimulation applied to the inputs. Requirements commonly specify admitted bounds of transient signals at the system interface. Therefore, the objective of this extended flow is to identify the impact factors have on the response and to find safe bounds for it, considering that the response varies in time. In addition, turning the worst-case predictions into graphical representations of margins for signals is attractive to system-level simulation practitioners, especially in the analog mixed-signal world, as opposed to extensive statistical measures for the probability of system conformance.

Therefore, the concept of a response is extended here to the set of time-value pairs of such a signal. With respect to the time factor t, the sampled response r represents a set of highly correlated static responses, because they are values of the same signal, influenced by the same set of factors. Ideally, the factors with major impact remain the same over the time interval under study. But since different events occur at different times, not only the constant term c_0 of the polynomial metamodel, i.e. the value when all factors are zero, modifies in time, but in reality so can the effects of factors and their interactions.

Basically, the experimental flow as introduced before is run in a loop, on the records of a transient response. The main challenge to address is the problem dimension, since a signal can exceed 10^6 time samples recorded during a standard simulation. The rate of result analysis must be obviously reduced, otherwise the impact of postprocessing on the performance becomes unacceptable. The analysis is executed in fewer samples and the metamodel representations are "interpolated" over these points.

4.3.1 Problem description

The problem can be formally described as: given a vector of factors f of length n and a response signal $r(f,t)$, the dependency on f and bounds of r with respect to f must be estimated. The effects of f can vary in time, and so can the factor sets which determine global extremes of the response. For each simulation, f is kept fixed and $r(f,t)$ can be recorded, at least for a set of time samples S.

It is assumed that the dependency of r on f can be approximated by a polynomial metamodel:

$$R(f,t) = c_0(t) + \sum_{o=1}^{2}\sum_{i=1}^{n} c_i^{(o)}(t)\cdot f_i^o + \sum_{j=1}^{n-1}\sum_{k=j+1}^{n} c_{jk}(t)\cdot f_j\cdot f_k$$

The assumption is checked for correctness after processing the simulation results, in the Metamodelling step. For simplification, the sample points are equidistant:
$S = \{t = p\cdot T; p = 1...s\}$, where s is the number of samples (the simulation duration is at least $s\cdot T$). Metamodelling must estimate for each time sample the set of coefficients $c(t) = \{c_0(t);\ c_i^{(o)}(t);\ c_{jk}(t)\}$, where $o = 1,2;\ i = 1...n;\ j = i...n-1;\ k = j+1...n$. The metamodel must be validated using the residual set, for each time sample. The relative residuals, i.e. normed to the maximum response variability, are analyzed. They also vary in time:

$$\epsilon(f,t) = (r(f,t) - R(f,t))/(max_f R(f,t) - min_f R(f,t))$$

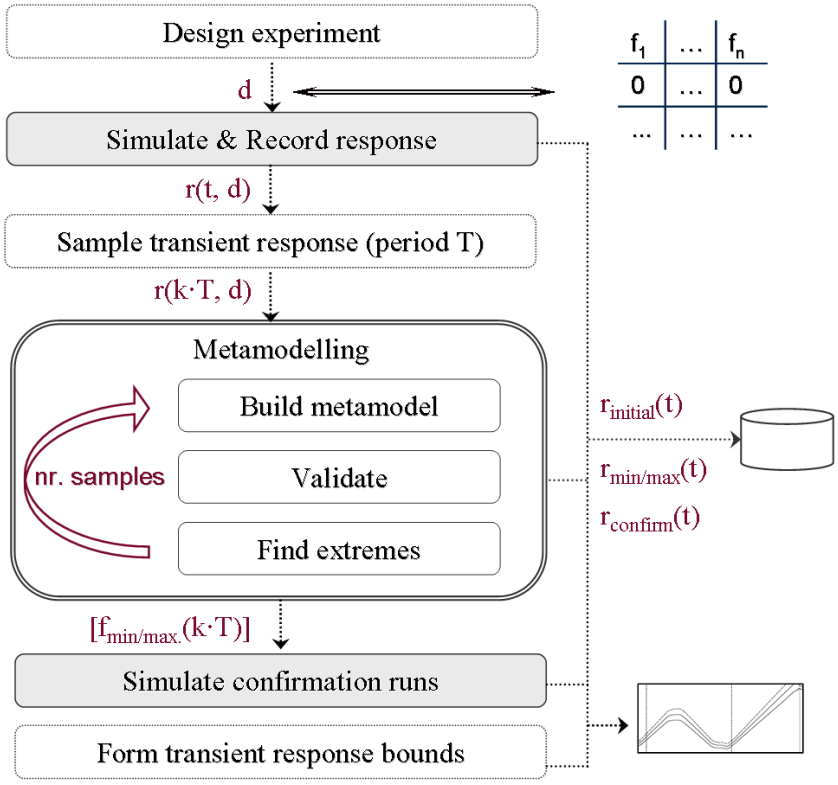


Figure 4.3: Transient response experiment flow

4.3.2 Flow

Figure 4.3 shows the experiment flow for a transient response.

Define the factor set The factor set is defined as described in Section 3.1.2. An experimental design is generated depending on the number of factors n which must be handled and the number of runs m which can be invested. The DoE matrix d of size $m \times n$ is loaded to be simulated. The first run is always the center point of the factor space ($f = 0$). This run provides a reference for each time sample.

Simulate the experiment The set of runs is simulated. The response signal is recorded for the complete duration of each simulation run, with a predefined time resolution.

Sample the response The traces are loaded into the postprocessing algorithm and a matrix of samples is formed: $[k \cdot T;\ r(k \cdot T, d(1,:)) \ldots r(k \cdot T, d(m,:))],\ k = 1 \ldots s$, where $r(k \cdot T, d(i,:))$ is the response result for run i, at time sample $k \cdot T$. $d(i,:)$ is the row i of DoE matrix, i.e. with the factor settings from run i. The failed runs are filtered out and the outliers are replaced with NaNs. When too many runs failed, the controller aborts the analysis and optionally reruns with reduced factor ranges. The outliers, on the other hand, are removed only for the respective time samples. For instance, a response signal which ramps-up and is triggered at different points

in time can present such outliers for sample times around the trigger event. This restriction is necessary to keep the response assumptions valid (discontinuities are removed).

Metamodelling For each time sample, the vector of results $r(k \cdot T, d)$ and the matrix d are used to:
Build the metamodel: Regression analysis extracts least square estimates for the coefficient set $c(k \cdot T)$.
Validate the metamodel: The metamodel must be characterized in terms of fitness on data, in order to be used for interpretation of effects and response prediction. For this, the residuals are analyzed according to the criteria defined in Section 3.3.2.2. Optionally, some runs can be invested to optimize the coefficient set, by performing iterative regression and validation. Samples where the metamodel is unfit are removed from the analysis. When they are too many, the controller aborts the analysis and optionally redesigns the experiment for a higher metamodel accuracy. Important factors and interactions, and the way they vary in time, can be identified by the estimated coefficients, and separately plotted for interpretation.
Find the metamodel extremes: Details on this step were explained in Section 3.4.3. Optimization strategies vary depending on the type of effects. The output of this step is the pair of estimated factor sets for extreme values of the metamodel, together with the specific extreme response values:

$$f_{min/max}(k \cdot T) = f(k \cdot T)_{R(f,k \cdot T)=min/max}$$

Simulate confirmation runs Simulations can be run to confirm the predictions with respect to the response's extremes, for time samples of interest. Only the significantly different predictions are run, i.e. only those runs, for which factor sets are far enough from each other. Results show that the impact of factors and, consequently, the factor set which determines a response extreme value, vary in time.

Form bounds for the response Finally, bounds for the transient response are "assembled". While metamodelling interpolates the response with respect to the factor set, the sampling in time and assembling of the response bounds performs a time interpolation of the response dependency on the factors. The confirmation runs completed by the predictions are used for the samples $k \cdot T$. Between these samples, the response can be predicted by extending the dependency on the factors from the closest time sample. Still, the constant term is adjusted using the value $r(f = 0, t)$. Safer response margins can be obtained by corrections based on the residuals (as center value or maximum).

Finally, the controller must report the estimated metamodels as well as the predictions for extreme response values. Assembled response bounds and results of initial or confirmation simulation runs are also stored. Additionally, the resulting performance as described in the following is reported.

4.3.3 Performance evaluation

The experiment time is

$$\tau = (m + conf) \cdot \tau_{sim} + s \cdot \tau_{regress\_validate} + s_{fit} \cdot \tau_{predict}$$

where s is the number of time samples, m is the number of runs of the DoE. $conf$ is the number of confirmation runs. s_{fit} is the number of time samples where the metamodel is found fit. m is the required number of runs for usual, fixed-level DoEs (the dependency on the number of factors is visible in Figure 3.15). The error rate of the experiment is given mainly by the results of residuals' evaluation. The number of significantly different sets of effects and extreme response predictions shows how complex the response dependency in time is.

Section 5.2.7 presents results of such a performance evaluation. The discussion paragraph 6.2 concludes that a careful monitoring of the postprocessing steps can lead to an efficient overall performance for transient signals.

5 Results

The presented flow and alternatives are demonstrated on selected automotive systems. A selection is made out of the existing approaches introduced in Chapter 2. Their implementation is first described. Then, the first case-study, i.e. an ECU designed for window lift applications, is validated by fixed-level and RFDoEs, as well as by transient response analysis. Afterwards, a system designed for airbag control applications is analyzed, with more focus on RFDoEs. Alternatives are tested on both systems and compared to the proposed solutions. The next chapter summarizes these results and makes comparisons between them.

5.1 Implemented alternatives to the DoE flow

Out of the set of alternatives presented in Chapter 2, the ones worth implementing (and marked so in Figure 2.1) are considered because of a higher potential to solve the issues we face. To implement them and reuse the experimental framework in place, only the simulation controller is modified. Some of the alternatives require only small modifications of the experiment planning algorithms.

5.1.1 Directed test methods

They require no adaptive test, thus there is no feedback of results to the instance planning the simulations. In addition, no complex postprocessing steps are required (no metamodelling, no response prediction or estimation of the worst-case). Sensitivity analysis can be performed only by simple computations on the response variance, while the worst simulated case is used to estimate the worst-case.

Trial-and-error

An example is the nominal value simulation, or best guesses on where the worst response case would occur. Their implementation is simple, but application dependent, and requires knowledge of the system.

One-at-a-time approach needs to successively vary each factor and keep the others at fixed levels, e.g. 0. As compared to the custom DoE introduced in Section 3.2.4.1, the planning is similar,

the effort to implement is comparable, and the number of runs is identical. The results of the one-at-a-time alternative can easily be derived from the ones of the custom DoE.

This class of methods is easily implemented by loading the factor settings to simulate in the DoE matrix, and feeding it to the controller. Nominal value simulation is a vector of 0-s for the factors, best guesses are manually set, while one-at-a-time approach, is realized by a basic loop which increments the current factor level and launches simulations, similarly to grids.

Exhaustive search

This method exhausts all combinations of factor levels, similarly to a multivariate grid. When factor ranges are continuous, it is not feasible. Still, they can be discretized, and then a number of $\prod_{i=1}^{factor\_number} number\_levels_{factor\_i}$ runs is needed. Since the dimension of the factor set often exceeds 10, the only acceptable case is the full set of corners, where 2 levels are invested per factor, just as in a 2-level Full Factorial. However, no metamodelling is involved, because there is no prediction or confirmation of the worst-case in directed test methods. It is easily implemented by simulating all corners, i.e. a DoE matrix of all vector combinations with elements in the $\{-1, +1\}$ set.

5.1.2 Worst-case direct search

A similar method to the iterative gradient-based search proposed in the approach (Section 4.2) is implemented. The iterative nature of the search is kept, but the following settings are different:

- as opposed to initial method, which is performed on the screened set, the direct search is applied on the complete initial factor set.
- the direct search uses the center as the starting point, while the initial method starts from the worst-case found after the sequential DoE.
- a bigger search step is used.
- the stop criteria is modified: a bigger change in response is expected before stopping and more runs are allowed before stopping.

Results can be found on the airbag case study, as compared to the proposed flow, in Section 5.3.6. The flow is found efficient only as initially proposed in Section 4.2 of the alternatives.

5.1.3 Random test methods

Random test is performed here as alternative to the proposed RFDoE. The conceptual differences between random test methods (Monte-Carlo) and RFDoE were explained in Section 3.2.5. They involve similar planning and the same number of runs, but basic random test methods do only randomization. No correlation or de-correlation of factors or measurements of the distances between samples in the factor space are performed. Just as in trial-and-error methods, no analysis of factors' effects, no response characterization and no worst-case prediction are performed. The implementation is quite simple: the random generator is called independently for each factor, passing the statistical distribution of the respective factor. The sets of random values are loaded into the DoE matrix, which is then simulated and plotted together with the results.

Results which compare Monte-Carlo against proposed approaches are presented in Section 5.2.3, but also shortly in Tables 5.6 and 5.12.

5.1.4 Genetic algorithm

The basic necessary concepts were briefly introduced in Section 2.5. GA is implemented to be compared against the proposed metamodel-based methods, in terms of worst-case results and general performance on the systems under study. The implementation uses a population size of: $N = ct \cdot n$, where n is the number of factors, and ct is chosen minimum 2 (e.g. 5). A number of generations of minimum 10 is used, therefore a 10-factor set would typically involve 500 runs, i.e. comparable to a LHS design. The mutation and crossover probabilities are varied according to how much the GA should span the domain of interest when generating the next population.

Judging from the results, response surfaces are found appropriate metamodels for the responses of interest. That is why the current GA implementation is tested initially on polynomial responses. Similarly to Figure 4.2, Figure 5.1 plots GA results evaluated on a 2-factor polynomial, with a superimposed noise (a random term). Although some progress towards the response extreme (maximum in this case) exists, the random changes applied from a generation to the next do not help to approach the real maximum. Many runs must be invested to reach a rather poor final response: $N = 10$, $G = 10$, i.e. 100 runs.

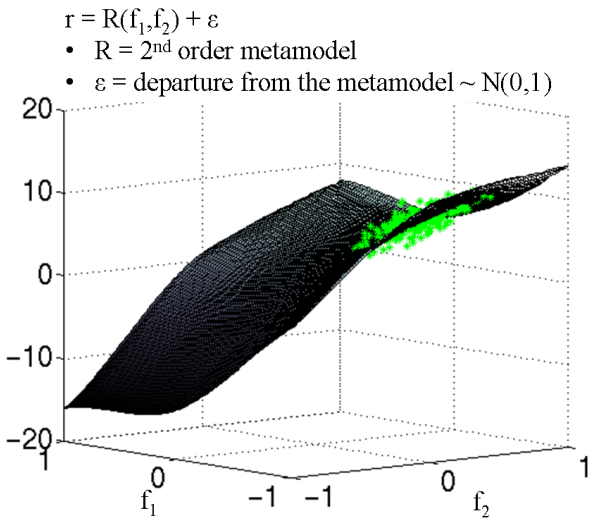


Figure 5.1: GA search example

The algorithm is tested for applicability on the real system, in Section 5.2.6. Because of the same reason, i.e. use of random sources in the next population generation, a low performance is recorded.

5.2 Window lifter system

The first case-study is an automotive ECU designed for window lift applications. The experiment must evaluate the system in the complete application context, so the DUT consists of the ECU and the electro-mechanical system driven and supervised by the ECU. Therefore, the system includes mechanics, analog and digital electronics, as well as software. It is a heterogeneous system and presents multiple sources of variability which introduce multi-nature factors.

5.2.1 System description

Figure 5.2 represents the basic structure of the electro-mechanical subsystem. The ECU controls a DC motor through relays. The motor acts on the gear box and a gear rack finally drives the window. A Hall sensor provides the ECU with the speed and direction of the motor, in the form of electric signals, for window position track.

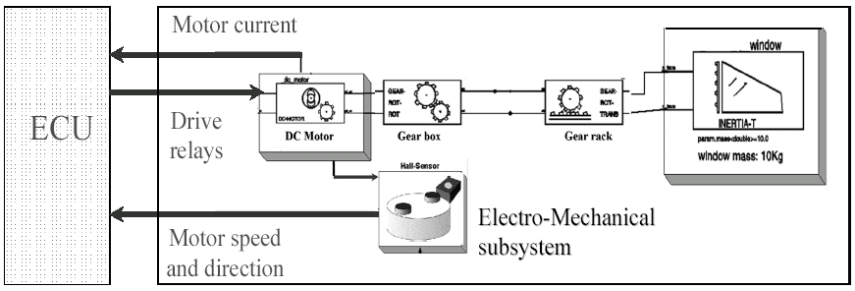


Figure 5.2: Window lifter electro-mechanical subsystem

The ECU structure is visible in Figure 5.3. The central digital element is a microcontroller

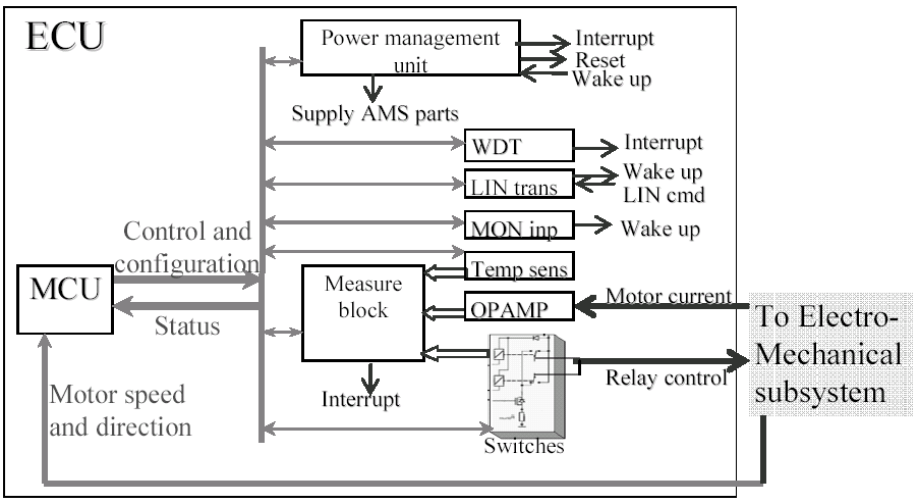


Figure 5.3: Architecture of the window lifter ECU

subsystem (MCU) compatible to the standard 8051 core. Within this application context, the LIN transceiver acts as an interface to provide the MCU with commands to control the mechanical load. Two low-side switches (LSS) with protection against short circuit and open load are driven for the relay control. The motor current is measured across a shunt resistance and transferred by

an operational amplifier (OPAMP), to a measurement interface (MI), consisting of 8-bit ADCs. The diagnosis element is a processing unit contained by the measurement block, which receives information from other sources which need supervision as well, e.g. the temperature sensor or the power supply. The processing unit signals the MCU if error conditions exist (overtemperature, overvoltage, overcurrent). The MCU interprets these signals and controls the system accordingly. Such protection against overcurrent is needed when an obstacle blocks the window (also referred as anti-pinch response). The power management unit is responsible for internal and external power supply and supervision. Low power modes are available, from which wake-up is possible via the LIN transceiver or monitoring inputs. A watchdog timer (WDT) supervises periodic trigger inputs from the MCU.

The DUT model The DUT was modelled using SystemC and its extensions SystemC-AMS and SystemC TLM1.0. The available models of computation and communication (Discrete Event, Timed Data Flow, TLM) were used to model the ECU and the electromechanics. An algorithmic model of the application software is used for the MCU, which was first validated against the results of an Instruction Set Simulator. The electromechanics model uses a nonlinear proprietary extension of SystemC-AMS, presently under evaluation. In this context, complete application scenarios can be simulated. Several test cases were verified: window control, with position track and anti-pinch response; tests of protection features e.g. overtemperature; power-up and power mode management.

5.2.2 Responses and factors

Responses are first defined, considering the requirements and expected behaviour of the system. Then, factors are identified as sources of variability with potential impact on the responses. Their ranges are extracted from the specification, as explained in Section 3.1.2.

The first experiment is performed on the anti-pinch response. For safety reasons, it is necessary to ensure that the maximum force applied to the external obstacle is within a specified range, given the allowed variations in different blocks and operating conditions. Therefore it is chosen as experimental response. Other responses associated to this scenario can be identified and they all relate to the event of obstacle occurrence: maximum current reached at the ECU interface, delay in the ECU reaction, from the presence of the obstacle until the relays switch off, driven by the LSS.

A correct interpretation and filtering of the specification information are essential at this point. The initial set of factors is common to all responses, and is collected from parameters of the blocks involved in this scenario. Factors are extracted from the specification and transferred into a configuration file, similar to the one presented in Figure 3.5.

Figure 5.4 contains this information in a tabular form. Each factor has a nominal value (considered the central value of its range) and a tolerance. Optionally, a statistical distribution can be specified. Along with the name to use in all pre- and postsimulation steps, the path to the corresponding parameter in the DUT model must be specified. Another field is used to mark if the factor is active in the current experiment.

The first level of hierarchy of the test bench schematic is visible in Figure 5.5. The corresponding test case is implemented in the stimuli function, of the Cfg&Stim unit.

factor	symbol	description	nominal (‡	tolerance	path in model	active	distribution	correlations
	v_source	Supply voltage (for electronics) [V]	...	...				
					dut->i_window_lifter->i_pm	y	uniform	none
	battery voltage	Battery voltage for the mechanics	...	...	dut->vbat_mec	y	uniform	v_source*0.9
	shunt_resistance	Shunt resistance [mΩ]	...	...	dut->i_mech_subsystem->i	y	normal	none
	opamp_gain	OPAMP gain	...	...	dut->i_window_lifter->i_opa	y	uniform	none
	lss_sr	LSS Slew Rate [V/µs]	...	...	dut->i_window_lifter->i_lss	y	normal	none
	lss_delay	Low Side Switch delay [ms]	...	...	dut->i_window_lifter->i_lss	y	uniform	none
	mi_delay	MI delay [ms]	...	...	dut->i_window_lifter->i_me	y	normal	none
	lin_delay	LIN delay [ms]	...	...	dut->i_window_lifter->i_tlm	y	uniform	none
	adc_delay	ADC conversion delay [us]	...	...	dut->i_window_lifter->i_sdf	y	uniform	none
	dc_motor_kt	Motor torque ct. [V/(rad/s)]	...	...	dut->i_mech_subsystem->i	y	uniform	none
	dc_motor_d	Motor damping coef. [N·m/(rad/s)	...	...				
					dut->i_mech_subsystem->i	y	normal	none
	dc_motor_j	Motor moment of inertia [Kg·m^2]	...	...	dut->i_mech_subsystem->i	y	normal	none
	dc_motor_r_wind	Motor winding resist. [Ω]	...	...	dut->i_mech_subsystem->i	y	uniform	none
	dc_motor_l_wind	Motor winding induct. [H]	...	...	dut->i_mech_subsystem->i	y	normal	none
	gear_rack_g_r	Gear rack ratio	...	...	dut->i_mech_subsystem->i	y	normal	none
	gear_rack_radius	Gear rack radius [m]	...	...	dut->i_mech_subsystem->i	y	uniform	none
	gear_box_g_r	Gear box ratio	...	...	dut->i_mech_subsystem->i	y	corner_peak	none
	inertia_mass	Window mass [kg]	...	...	dut->i_mech_subsystem->i	y	uniform	none

response	name	description	expected	expected t	monitor function call	measu	testcase
	antipinch_delay	ECU delay (obstacle hit to LSS off)	...	...	meas_delay(window_pos, (	y	antipinch
	antipinch_force	Maximum force on obstacle [N]	...	...	meas_max(dut->i_mech_s	y	antipinch
	antipinch_current	Current switch off value [A]	...	...	meas_max(ecu_current_i)	y	antipinch
	antipinch_w_pos	Window stop position [m]	...	...	meas_max(window_pos)	y	antipinch

Figure 5.4: Window lifter DUT configuration file

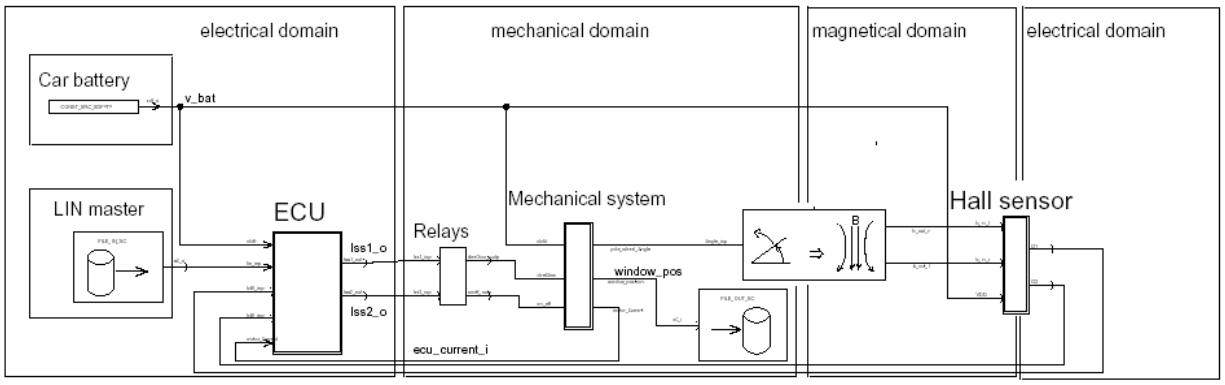


Figure 5.5: Window lifter testbench

Nominal value simulation

Results of the nominal value simulation are presented in Figure 5.6.

The close window command is processed and the MCU controls the LSSs to drive the DC motor. When an obstacle blocks the window, the force developed by the DC motor increases and so does the current. The MI converts the amplified current value and the postprocessing block flags the error condition to the MCU, which switches the LSSs off. For functional verification purposes, relevant internal and external signals were traced, and correct timing and event sequences were checked. These are not performed in the experiment, however, since they are not necessary, cannot be checked for each run and would introduce a decrease in the simulation performance. Instead, the response results are checked against the expected range, to make sure the system still has the same functional behaviour.

In the following, different experimental approaches are demonstrated, on different factor sets. The next two sections present experiments on the first factor set, which are focused on probability

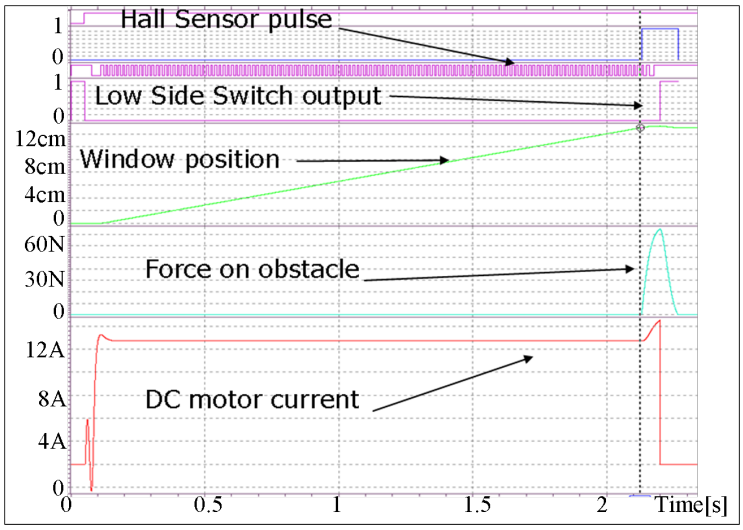


Figure 5.6: Nominal value simulation of the antipinch test case

distributions and correlations in factors. These are best applied on RFDoEs, but fixed-level factor DoEs are also implemented for comparison. The second factor set analyzed in Sections 5.2.5 shows how simple responses can be and that experiments with minimum resources can reach the objectives. The third factor set is analyzed in Section 5.2.6 to show a sequential approach, suitable for more complex effects, and when a deeper worst-case analysis must be implemented. Finally, the extended analysis for transient responses is demonstrated in Section 5.2.7.

5.2.3 Results of the random factor DoE versus Monte-Carlo

The experimental methods are first demonstrated on a reduced 4-factor set (factor set 1). The target is to show the applicability of experimenting with factors statistically distributed and correlated as explained in Section 3.2.3. Basic Monte-Carlo and RFDoEs, as introduced in Section 3.2.5, are tested and analyzed as detailed in Section 3.4.4.

Experiment

The responses in the first experiment are the maximum force on the obstacle and the maximum switch-off delay. The selected factors are: $Vbat$ (supply for the overall system), Vs (supply for the electronics), the $OPAMP\ gain$ and the $LSS\ delay$ (Figure 5.4). Vs is strongly correlated to $Vbat$ because the supply voltages have a common origin, even if the electrical components which interface the two power supplies introduce additional variation and weaken the correlation.

500 simulations were run, with factors normally distributed, and the two responses were recorded and mapped to the factor sets. This corresponds to a distance in the factor space of less than 0.1. The distributions are generated as described in Section 3.2.3. The plot matrix in Figure 5.7 shows scatter plots for each pair of factors, the responses with each factor, histograms for each factor and for the responses.

Some interesting effects can be observed: the force increases with $Vbat$, for smaller values, probably because the DC motor develops more force against the obstacle as the supply voltage increases.

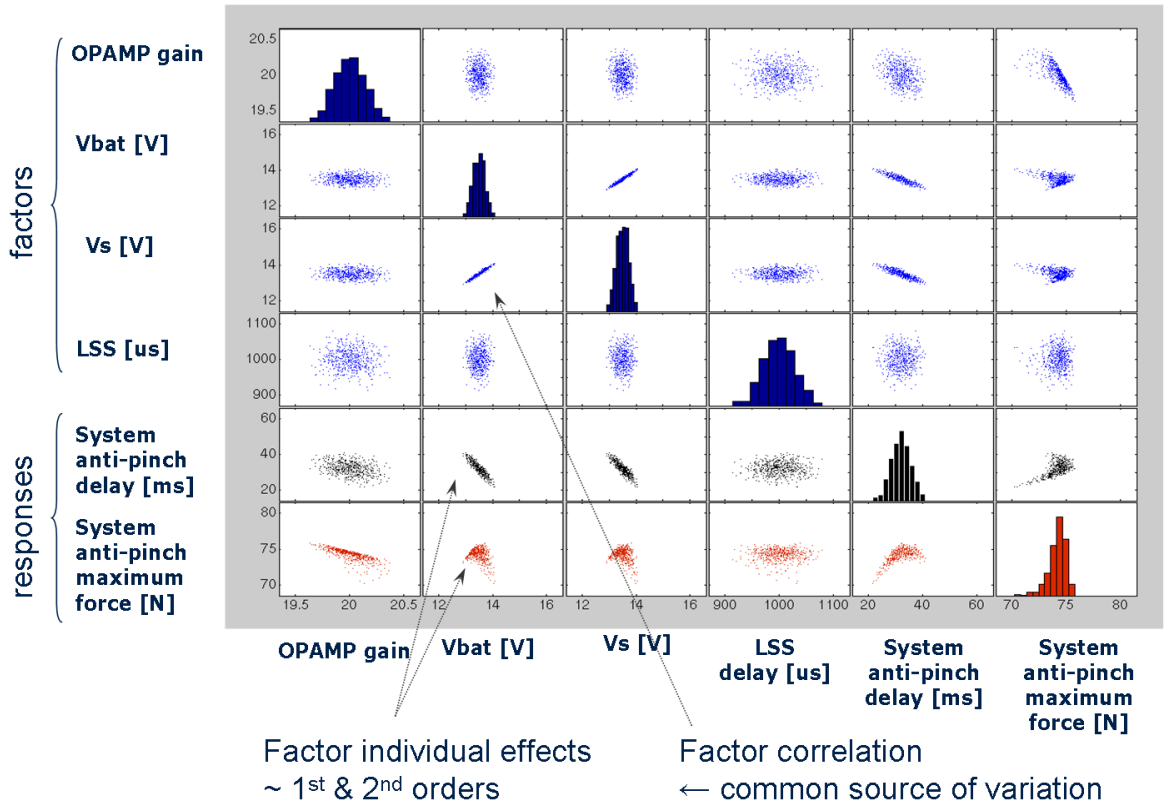


Figure 5.7: Plot matrix of an RFDoE on window lifter responses

But for higher values, the force tends to decrease. The reason could be an ability to stop the DC motor faster and finally a smaller force. The force decreases with the *OPAMP gain*. That is because the over-current condition is triggered faster for higher gains. But the *OPAMP gain* effect on the delay response is masked by more important ones. The force should increase with the *LSS delay*, but the effect is also masked by others (the maximum variation of the *LSS delay* introduces less variability than other factors in the total reaction delay, thus in the force). The main effect visible in the delay is a decrease with high supply voltages, because of a bigger speed of reaction of the system.

Factor effects

While basic Monte-Carlo (i.e. random test) stops here, the RFDoE continues with a sensitivity study to confirm the observations detailed above and with the worst-case analysis. The correlation extracted from the matrix plotted in Figure 5.7 can be used to characterize the factors' effects, as explained in Section 3.4.4.1. The table below contains the linear correlation coefficients, with the color corresponding to the plot matrix.

A test of the hypothesis of no correlation between the responses and the factors is performed in order to judge on significant factor effects, as related to each other. The 2-row matrix below is the resulting probability of the test, and indicates that the single insignificant factor is the *LSS delay*.

1.00	−0.04	−0.05	−0.03	−0.36	−0.76
−0.04	1.00	0.98	0.00	−0.91	−0.32
−0.05	0.98	1.00	−0.01	−0.89	−0.31
−0.03	0.00	−0.01	1.00	0.01	0.01
−0.36	−0.91	−0.89	0.01	1.00	0.62
−0.76	−0.32	−0.31	0.01	0.62	1.00

0	0	0	0.682
0	0	0	0.743

A mapping to the effects visible in the matrix plot can be established. The correlation coefficients factors-response quantify factor effects, thus allow comparisons. These linear correlations estimate the main effects of the factors. Their signs and magnitude, as referenced to each other, can be interpreted for sensitivity analysis, similar to the one previously described, but more objective. The independent factors have a correlation coefficient smaller than the 0.05 threshold, and the correlated factors have a coefficient close enough to the one initially specified.

Response distribution and worst-case prediction

Analysis of the distribution for the force response first fits a function on the empirical PDF. Figure 5.8 shows a fitted nonparametric distribution, for the maximum force response.

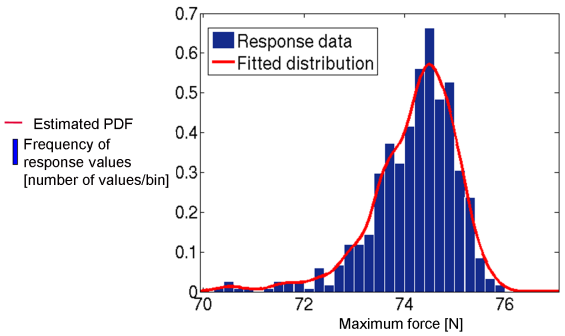


Figure 5.8: Fitted response distribution

The fitness is given by a residual probability of $mean(res\_prob) = 0.077\% < 0.1\%$, i.e. small enough. The response range is estimated to $[69.45...76.61]N$ with a threshold probability of $\epsilon = 1.0e - 6$. To conclude, distributing and correlating factors according to their statistical properties "naturally" covers the multi-dimensional verification space. But RFDoEs continue the Monte-Carlo experiment by extracting correlation as a measure of factors effects and space coverage. Then, the response distribution is estimated and used to predict the worst-case.

5.2.4 Results of the Central Composite DoE versus alternatives

The first part of this section shows results of fixed-level and random factor experimental designs. The second part applies the custom DoE and the third compares the results.

CCD DoE, RFDoE and Monte-Carlo

$Vbat$ is removed and $Rshunt$, an independent factor, is introduced. A CCD DoE is simulated, which requires only 25 runs. The quadratic regression model is estimated, and it locates a pronounced 2^{nd} order effect of Vs and important interaction effects. The coefficient set quantifies effects for the factor set:
$[F_1 = Vs; F_2 = Rshunt; F_3 = OPAMP\ gain; F_4 = LSS\ delay]$. The metamodel has the constant term c_0=75.71, the vector of linear effects
c_l=[1.04 -1.44 -0.67 0.11]. The matrix with interaction and quadratic effects c_{ij} is visible below. The coefficient estimates are validated against the regression results of a 500-run RFDoE.

-2.040	-0.165	-0.375	0.060
-0.165	0.010	-0.003	-0.020
-0.375	-0.003	-0.090	-0.015
0.060	-0.020	-0.015	-0.010

Figure 5.9a shows an interactive plot of the regression model. Such a plot updates the predicted response when changing the factors levels (indicated by the vertical lines) [50]. This makes the 2-dimensional graphical view possible. 95% confidence bounds for the response are also extracted. The residuals are computed and analyzed as described in Section 3.3.2.2. Their normal probability plot (Figure 5.9b) indicates that they are approximately normally distributed with mean 0.

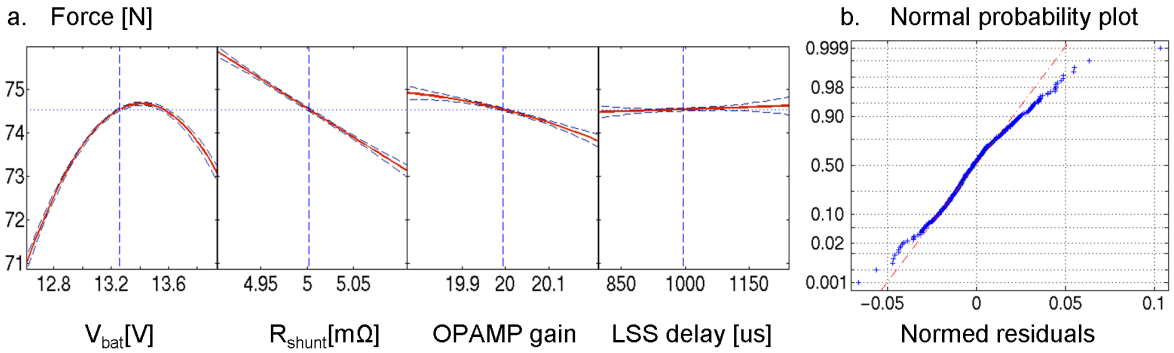


Figure 5.9: a. Regression results on the RFDoE
b. Normal probability plot of residuals

To compare the initial DoE and the basic Monte-Carlo, Figure 5.10 plots the results of the Monte-Carlo simulations and the metamodel resulting out of the CCD DoE. The worst found cases are marked on the figure. For the Monte-Carlo experiment, this is the worst simulated case. For the CCD DoE the worst-case is first predicted as detailed in Section 3.4.3, then confirmed by simulation.

To conclude, CCD DoEs can be quite powerful, because they extract sufficient information out of much less data, i.e. even after 20 times less runs than Monte-Carlo. Regression on the results of a considerable number of runs indicate adequacy of regression metamodels for the major effects in this system. However, statistical properties of factors and characterization of the response distribution can be performed only with RFDoEs.

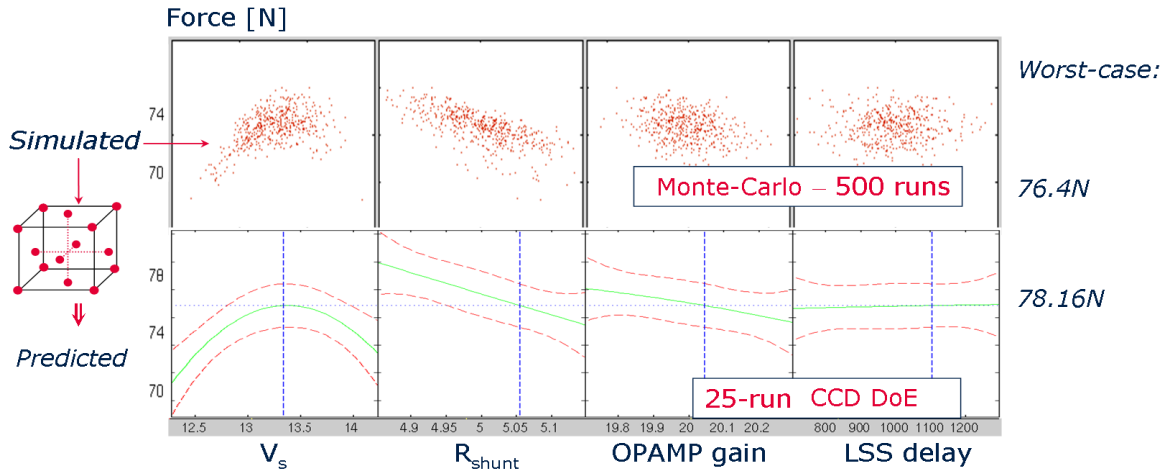


Figure 5.10: RFDoE results versus CCD DoE regression results, on the maximum force on the obstacle

Custom DoE

The previous results pointed Vs as an interesting factor, due to 2^{nd} order effects, as well as interactions with $OPAMP\ gain$ and $Rshunt$. The custom implementation of a multi-level factorial introduced in Section 3.2.4.1 is tested next. A special focus is put on the main factor, and corresponding analysis of results and estimation of effects are performed: Vs is the main factor, with 10 levels over its range. Simulation sets, with the other factors (=factor set) set to corners, are run at each main factor level. Results show an interaction between the main factor and the factor set, plus a non-monotonous response, over the main factor range.

The force is interpolated over main factor levels, thus assumed piecewise linear, for each set. Figure 5.11 shows that interaction exists: plots of different factor sets are not parallel and the worst-case corresponds to different levels of the main factor, depending on the choice for the factor set. The response variability on $Vbat$ is different for different sets, i.e. another sign of interaction.

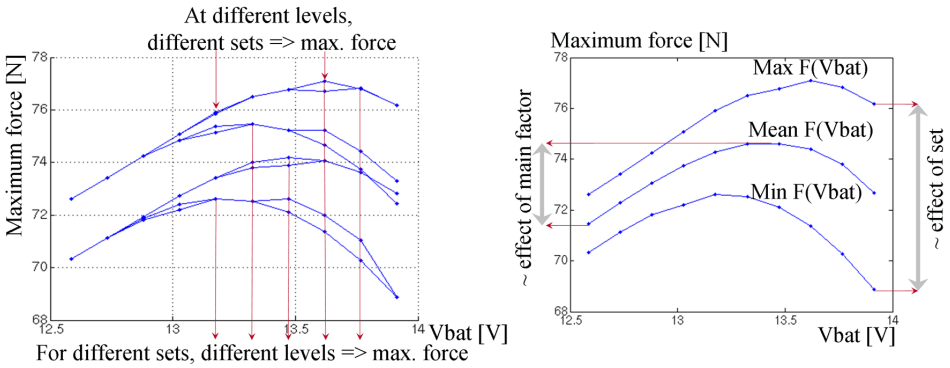


Figure 5.11: Interpolated response for the custom DoE

The effects are computed as explained in Table 3.3. Estimates are summarized in Table 5.1. Similar conclusions can be drawn based on the interpolated $Force(index\ of\ factor\ set)$.

Effect estimates[N]		*Vbat*	*Rshunt*	*OPAMP gain*	*LSS delay*
Main factor effect	*mean (std_sets)*	1.27	1.00	0.61	0.05
Factor set effect	*std (mean_sets)*	1.62	1.50	1.7703	2.24
Interaction	*std (std_levels)*	0.53	0.06	0.22	0.02

Table 5.1: Effects in the custom DoE

Comparison

The worst-cases and the number of runs for tested experiments are presented in Table 5.2. The response's mean and standard deviation are also included.

Table 5.2: Relative improvement in the response variability

Method	*Number of runs*	*Worst-case response[N]*	*Mean response[N]*	*Std. dev. of response[N]*
Monte-Carlo	500	76.41	74.30	0.92
Custom DoE	80	77.09	73.49	1.96
CCD DoE	25	78.16	73.99	2.22

Monte-Carlo is therefore the poorest as number of runs, worst-case found as well as standard deviation. The CCD DoE has the highest performance, obtained with 20 times less runs, because of the well-fitted quadratic metamodel. The custom DoE also detects the non-monotonic response dependency on the main factor, as well as the interaction effects, but has no predictive nature. Therefore, its worst-case estimate is one of the initially simulated points, and is poorer the metamodel's confirmed prediction.

To conclude this first part, the CCD DoE reveals effects essential for a proper sensitivity analysis and response extremes better than the alternatives which require more runs: random test method and than the custom DoE. The custom DoE can locate higher order interactions, and approximate the response with piece-wise linear functions on important factors. RFDoEs contribute by a better coverage of the factor space with properly modelled correlations and distributions, both for the factors and for the response.

5.2.5 Comparison of Fractional Factorials

R3FF DoE is demonstrated next to be efficient when factor effects are quite simple. The experiment studies the maximum force response, and uses a 10-factor set (factor set 2). Figure 5.12 displays the interactive plot of the metamodel obtained after regression on the results of a 16-run experiment. The factor set under study is also visible.

Table 5.3 compares the simulation results for the worst-case predictions of the metamodels estimated after: the full factorial DoE, the R5FF DoE and the R3FF previously presented. It can be observed how significantly different the invested numbers of runs are, while the worst found case is the same. The factor set has insignificant interactions or quadratic effects, as compared to the main effects. Therefore the worst-case is estimated as one of the corners.

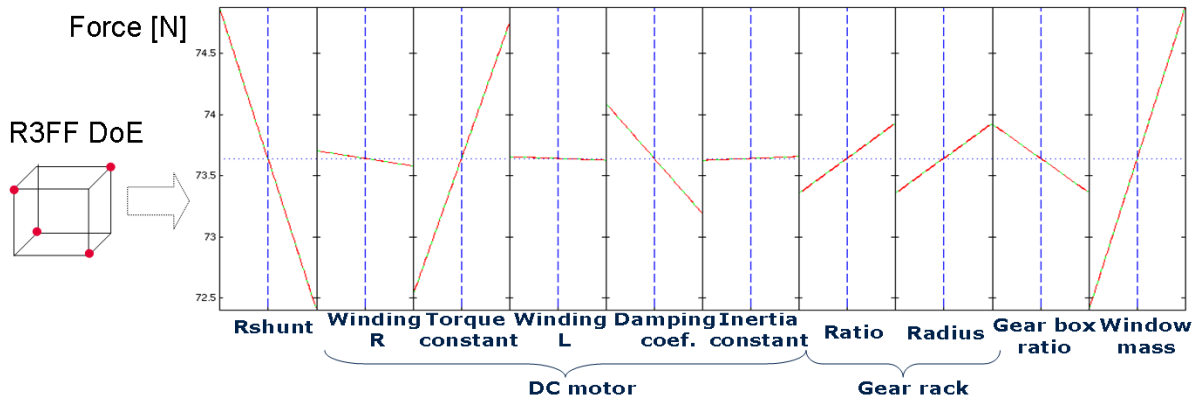


Figure 5.12: Individual effects on the maximum force response, after R3FF DoE

Table 5.3: Results of Factorials on the maximum force response

10 factors	*Full factorial*	*Fractional factorials*	
Number of simulation runs	1024	128	16
Simulated after prediction	78.488N	78.488N	78.488N

5.2.6 Sequential DoE and alternatives

More factors are considered in the next analysis, i.e. a 14-factor set (factor set 3) (visible in Table 5.4) from the set specified in Figure 5.4. A sequence of experiments is performed, in order to locate important factors and more complex effects.

Results of a screening experiment

The factor set is subject to a screening experiment. The same maximum force response is analyzed. To identify factors with little or no impact on the response, a R3FF DoE for this set of factors involves no more than 16 runs to extract preliminary main effects. The ANOVA method described in Section 3.3.1 is applied to identify factors statistically significant (Table 5.4).

The test compares the variance explained by factors relative the the total response variance (given by the mean sum of squares from the first column) with the left over variance that cannot be explained ϵ. The probability from the last column is the result of the hypothesis test of null factor effects. A rather permissive for p of 0.1 is used as decision threshold. Factors which are removed from the analysis, i.e. with $p > 0.1$, are marked italic in the table.

Fixed-level factor DoEs

The 10 remaining factors after the screening step are analyzed first in a R5FF DoE with 128 runs, to estimate main and interaction factor effects. The estimated metamodel is found not adequate after the residual analysis, because they are not approximately normally distributed. The left plot in Figure 5.13 indicates this lack of fit. Therefore a CCD DoE is built by augmenting the previous DoE with 21 additional points, resulting into a DoE with 149 runs. The simulation results fit a 2^{nd} order regression model. In this case the residuals are approximately linearly

Table 5.4: Window lifter ANOVA Table

Factor	*Mean Sum of Squares*	*Probability*
v_bat	1.024	0.063
r_sh	7.291	0.023
amp_gain	5.006	0.028
lss_sr	*0.008*	*0.524*
mi_delay	*0.134*	*0.170*
motor_kt	7.534	0.023
motor_d	3.738	0.033
motor_j	*0.018*	*0.408*
r_wind	*0.213*	*0.136*
l_wind	0.568	0.084
g_ratio	2.986	0.037
g_r	3.431	0.034
g_box_ratio	2.754	0.038
w_mass	0.827	0.070
ϵ	0.002	

distributed and have zero mean (Figure 5.13, right side). The model is adequate enough and can be used for response characterization and worst-case analysis.

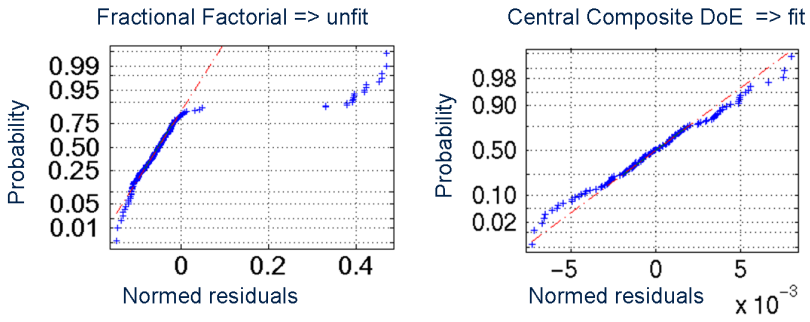


Figure 5.13: Residual analysis on the R5FF and CCD DoEs, on the maximum force response

Figure 5.14 shows the predicted response versus each factor, with the others fixed to 0. Main and quadratic effects are visible and can be interpreted, like for the plot matrix in Figure 5.7, but can also be used for response predictions. The new sources of factors, i.e. the DC motor, the gear rack, the gear box and the window, impact the force as well. E.g. a smaller window mass determines a smaller maximum force.

To view interaction effects, a plot like in Figure 5.15 is useful. E.g. a significant interaction exists between the supply voltage and the amplifier gain, because the impact of the voltage is affected by the level of the gain, and vice-versa. When the $OPAMP\ gain$ is high, the over-current condition occurs fast enough for the force to remain smaller. In this case, the supply voltage does not have a big impact on the force.

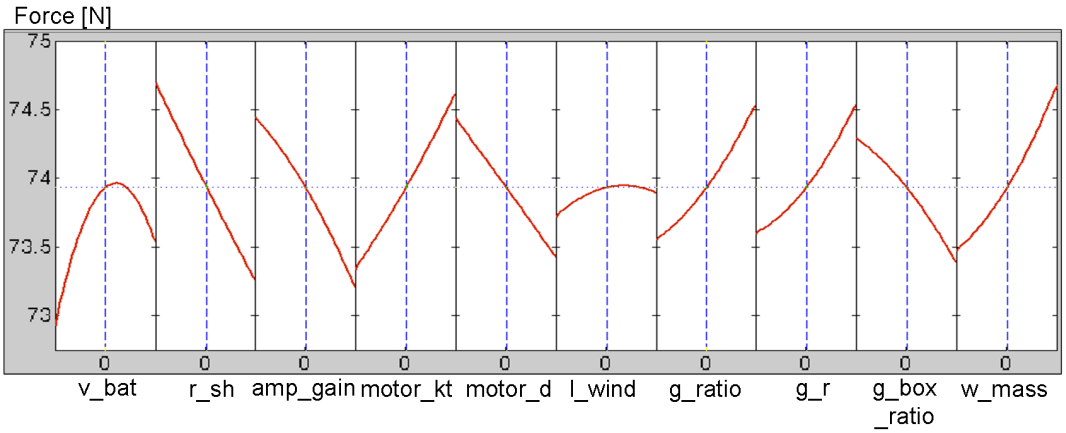


Figure 5.14: Individual effects on the maximum force response, after CCD DoE

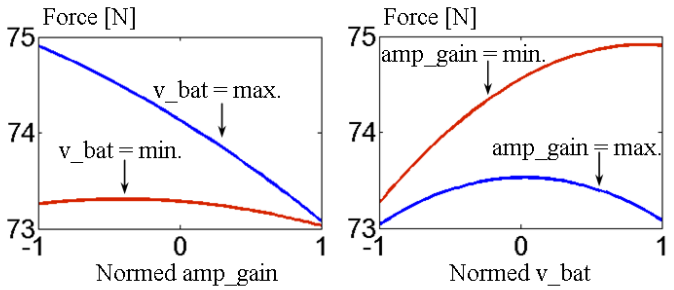


Figure 5.15: Interaction effects on the maximum force response, after CCD DoE

Worst-case analysis

Based on the metamodel previously estimated, the factor sets for extreme response values are predicted, as detailed in Section 3.4.3. Their prediction error, evaluated against the respective confirmation runs is less than 0.1%. The error is computed as:
$error = (simulated\_response - predicted\_response)/(max\_response - min\_response)$.
The results are summarized in Table 5.5.

Table 5.5: Worst-case maximum force response

Max. response [N]	*Normed factors*	*Min. response [N]*	*Normed factors*
Predicted	v_bat = 0.7387	Predicted	v_bat = -1
78.82	r_sh = -1	69.36	r_sh = 1
	amp_gain = -1		amp_gain = 1
Simulated	motor_kt = 1	Simulated	motor_kt = -1
78.77	motor_d = -1	69.37	motor_d = 1
	l_wind = -0.0168		l_wind = -1
Error	g_ratio = 1	Error	g_ratio = -1
0.06%	g_r = 1	0.01%	g_r = -1
	g_box_ratio = -1		g_box_ratio = 1
	w_mass = 1		w_mass = -1

Directed and random test alternatives

For comparison purposes, the exhaustive search directed test method is tested. The complete set of corners for the screened factor set is simulated, because the initial set of 14 factors is too large to allow exhaustive test. Monte-Carlo is also implemented, i.e. a set of 500 random runs on the initial factor set is performed. Table 5.6 summarizes the comparisons, in terms of number of runs and detected extreme response values.

Table 5.6: Comparisons of worst-case results

Experiment approach	*Number of runs*	*Extreme response values (Force [N])*	*Relative response variability*
Proposed sequential DoE	16+149 = 165	69.37 - 78.77	12.69%
Screening & all remaining corners	16+1024 = 1,040	69.37 - 78.69	12.58%
Monte-Carlo	500	72.15 - 75.85	5%

The extreme response values found with the sequential DoE are better than using the alternatives. In addition, the required number of runs is smaller with one, even 2 orders of magnitude. None of the alternative approaches characterizes the response. The DoE method measures effects by coefficients and can additionally predict the response between simulated points, with a small enough error (less than 0.1%).

Genetic algorithm alternative

The maximum force response is analyzed on the screened 10-factor set. The gradient-based search is performed after the sequential DoE, as explained in Section 4.2. It is compared against the results of the GA algorithm detailed in Section 5.1.4. The gradient-based search stopped after 64 runs. The GA used $G = 50$ generations, with $N = 10$ samples as the size of each population, i.e. 500 runs. Figure 5.16 shows all simulated samples for the gradient-based search, and only the maximum response from each generation, for the GA method.

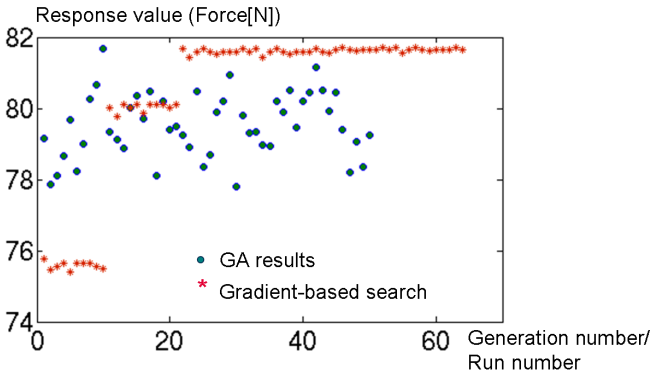


Figure 5.16: GA versus gradient-based search

Only the gradient-based search indicates an evolution in the proper direction. This can be explained by the fact that GA involves random generators in the search for the maximum, and

especially when applied on large factor sets, the method does not converge. The worst-case values also show a higher performance of the gradient-based search. Even after 500 runs, the GA finds a worst-case poorer than a gradient-based search of only 64 runs.

5.2.7 Results of transient response analyses

The flow proposed in Section 4.3 for worst-case analysis of signals over the complete duration of the test is demonstrated.

Experiment

The simulated test case is visible in Figure 5.17. For each window move command, the LSSs are controlled to drive the DC motor. At time 1.8 seconds, an obstacle in the window determines an increase in the force developed by the DC motor, thus the normal anti-pinch system reaction.

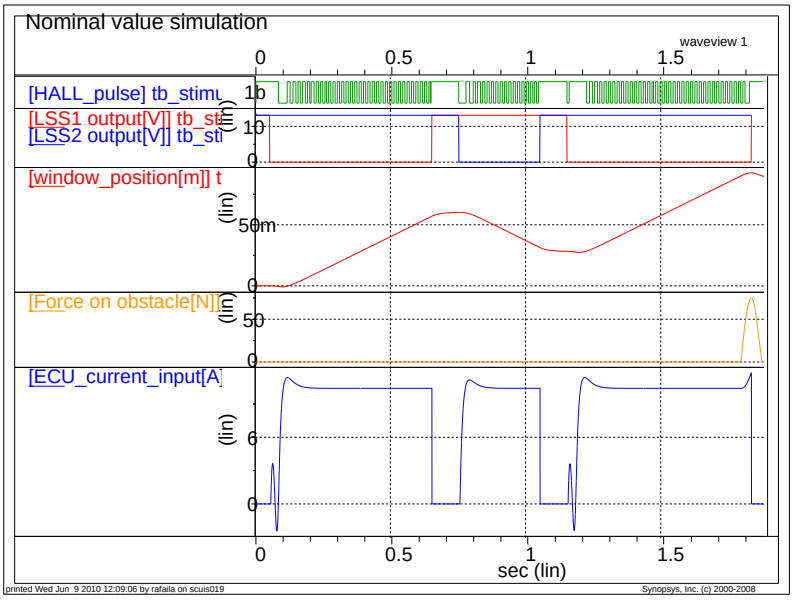


Figure 5.17: Simulated test case

Two responses are analyzed: the current at the ECU input and the window position ($r1$=current[A]; $r2$=window position[m]), as they introduced in Figure 5.4. The 10-factor set screened as shown in Section 5.2.6 is analyzed. The CCD DoE is augmented by a LHS DoE for more metamodel accuracy and for later validation against separate runs. A total of 200 runs is used.

Metamodelling

The initial response traces are sampled in $s = 100$ equidistant points. For each time sample, the data for each response is fitted and the metamodel is validated.

Figure 5.18 exemplifies a set of compliant residuals for $r2$, in an arbitrary time sample. The hypothesis of fit to a normal distribution of mean zero is passed. The correlation of residuals to the response values and the maximum residual are also tested. Samples where the metamodel is concluded unfit are removed from the analysis. This way, no extra effort is spent to predict the extremes for such samples.

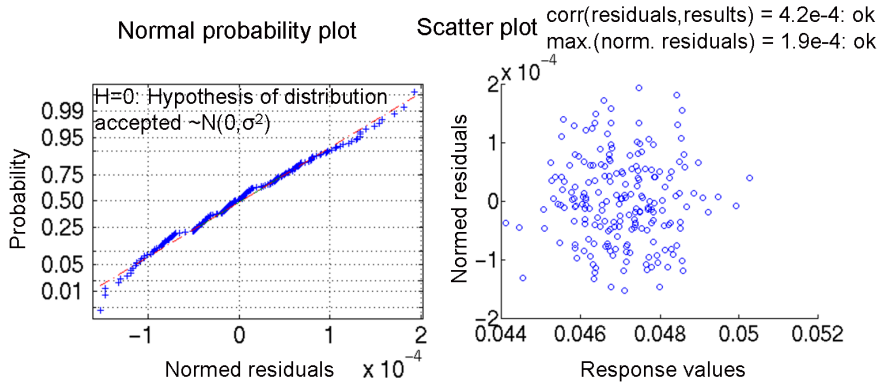


Figure 5.18: Residual analysis

To study factor effects, coefficients of the metamodels are studied at each sample time. Figure 5.19 shows the metamodel of response $r2$, plotted for a subset of 4 factors, when the others are set to 0, for 3 time samples. It can be concluded that the impact of factors varies in time. Different

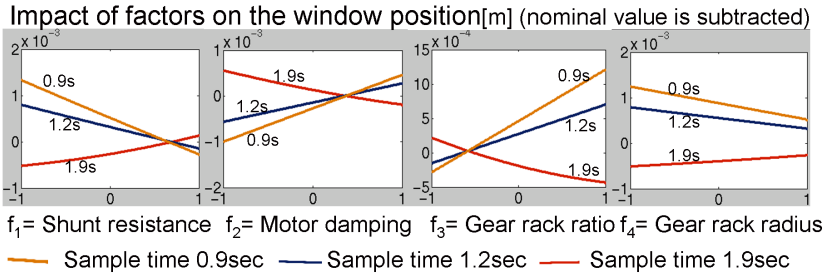


Figure 5.19: Factor effects at different time samples

time samples correspond to different factor effects, and consequently, to different predictions for the factor sets which determine response extremes. Some are exemplified in Table 5.7, for the window position ($r2$). The factor sets which were found are not only corners. This is justified by the presence of quadratic effects.

These steps are useful to identify and interpret effects which are otherwise hard to track, especially when they change in time. For instance, the effect of the *LSS delay* depends on the moment in simulation: the slower the switch, i.e. the higher the delay is, the lower the position at time 0.38 seconds is. At time 0.9 seconds, because of the two switching events which occur until then, the effect can become positive, by an increase in the response at an increase in the delay. The cumulation of factors' influences in time, combined with the mixed nature (interaction, quadratic) for these influences, can become rather complex.

Table 5.7: Worst-case predictions for $r2$

			$f(1)$	$f(2)$	$f(3)$	$f(4)$
Time	Min.	2.57cm	1	1	1	1
0.38s	Max.	2.82cm	-1	-1	-1	-1
Time	Min.	4.27cm	-1	-1	-1	-1
0.90	Max.	5.28cm	1	1	1	1
Time	Min.	7.84cm	0.35	-1	1	-1
1.76s	Max.	9.52cm	-1	1	-1	0.84
Max. residual				0.05		

Analysis of bounds for the responses

To analyze the worst-case over the complete simulation duration, the bounds for the two responses are formed as explained in Section 4.3. They are plotted in Figure 5.20. The figure also plots the resulted bounds after a similar analysis, but in fewer time samples ($s = 10$), and without interpolation.

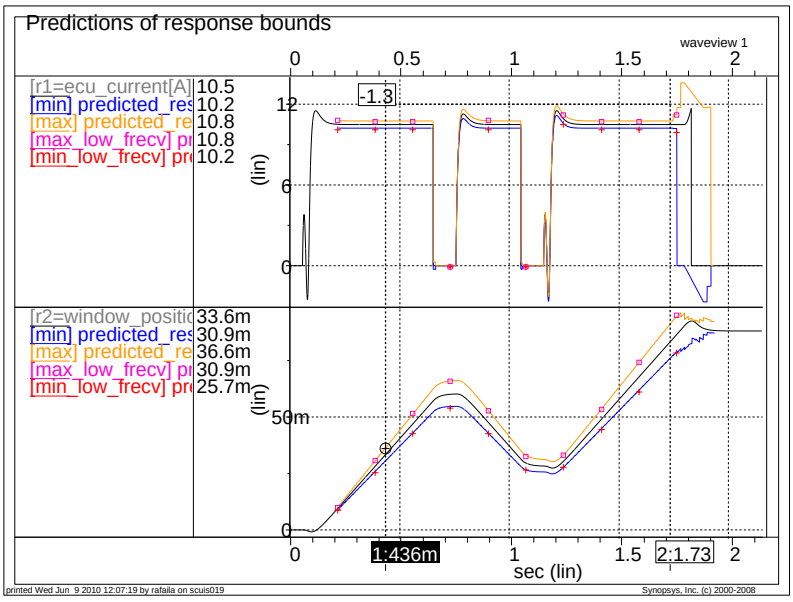


Figure 5.20: Transient responses

The large metamodel variance close to the end of the test is caused by the discontinuities in the response sampled in that interval. These are in turn effects of various switch-off times for different factor settings. When the obstacle occurs, factors which impact the DC motor speed, the window inertial effects and ECU block delays affect the reaction delay. This explains the outliers and lack of metamodel fit around the obstacle detection time.

Confirmation simulations are run for the worst-case predictions at different time samples. Only the significantly different predictions are run, i.e. only when the corresponding factor sets are far enough from each other. A minimum distance between factor sets of 1% is used to determine

whether to run confirmations. Figure 5.21 shows results of such confirmation runs, for different time samples, for the window position.

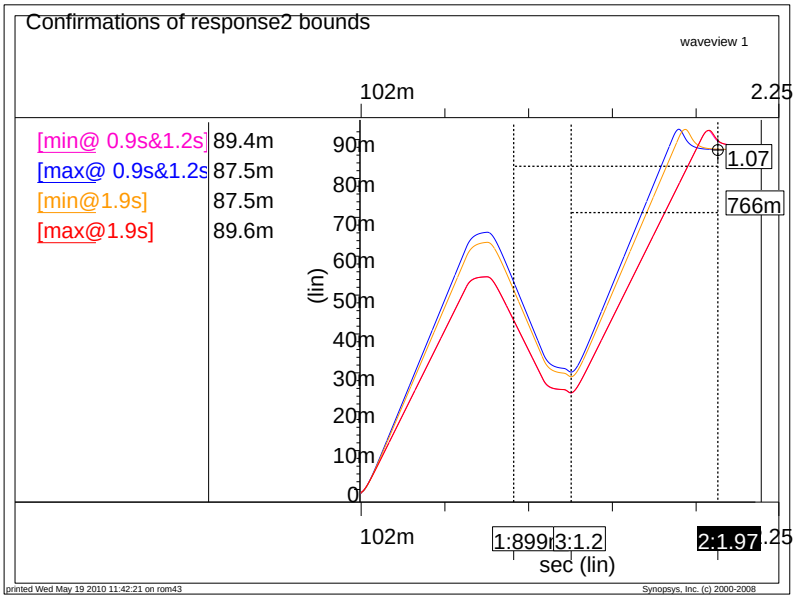


Figure 5.21: Confirmation runs

They confirm that the worst-case originates from different factor sets at different time samples, because the traces are intersecting. This also happens for the current, as factor sets which determine higher overshoots in current correspond to smaller settle values.

Performance

The performance is evaluated on the criteria introduced in Section 4.3. The experiment included a 200-runs initial experiment ($m = 200$), at a simulation duration of $\tau_{sim.} \approx 45$ *seconds* per run, corresponding to the 2 *seconds* of real time simulated in the test. None of them failed the functional test i.e. the response could be recorded for each.

Out of a total of $s = 100$ time samples, $r1$ results presented too many outliers in 5 time samples, while for 7 samples of $r1$ and for 8 of $r2$ a metamodel could not be fitted. These were caused mainly by the responses' discontinuities close to the end of the simulation, because of different switch-off times. For response $r2$, different sets of factor effects determined $conf = 18$ different worst-case predictions, both for response minimum and maximum. These were run for confirmation and output residuals smaller than 5%. The postprocessing durations are $\tau_{regress\_validate} \approx 4$ *seconds* per sample, while $\tau_{predict}$, of 8 *seconds* per sample, is invested only for the well fitted metamodels. The overall time is evaluated to $\approx 3h$, out of which the total simulation time is $\approx 90\%$ of the total sime, while the postprocessing time for fitting, validation and worst-case prediction for all samples is $\approx 10\%$ of the total time.

As a general note, a more complex system is likely to require a higher τ_{sim} (take longer time to simulate) and have more factors. Similar problems occur when the DUT model is more accurate: more details e.g. in the function or structure of a model, add factors and increase the simulation

time. But a higher number of factors n determines a higher number of experiment runs m, in order to fit a reasonable metamodel. $\tau_{regress\_validate}$ and $\tau_{predict}$ also increase with the number of factors to consider.

With respect to the influence of the number of time samples, a higher s reflects into a bigger post-processing duration. Still, this duration becomes significant only for a high number of samples. When the metamodels are unfit for too many samples, either a resampling or additional runs should be performed. Tracing should be done as little as possible, especially when the simulated duration is high, because it slows down simulation and would consume too much memory e.g. for more than 100 runs. Only the time samples of interest from the traced responses are loaded during postprocessing, because of the limitations in the run-time memory. To avoid oversampling or undersampling the simulated traces before the metamodelling step, an adaptive sampling step dependent on the rate of change in time of the response would also be an option.

To conclude, the experiment flow extended for transient responses can find how factor effects vary in time and predict safe margins for signals of interest. This does not involve more runs than in the case of static responses, i.e. with only one value per run. A careful monitoring of each step allows to control and limit the overall effort.

5.3 Airbag driver system

The second case-study is an automotive system designed for airbag control applications. Similar to personal restraint systems and ABS systems, airbag systems represent a major class of safety electronics. After they collect diagnostic data about the environment and the critical components of the vehicle, they must provide the driver with assistance and help to protect the vehicle occupants in the event of an accident. They face critical requirements with respect to fault tolerance and response timing, therefore the validation process must ensure a high degree of reliability.

5.3.1 System description

In such a system, the squibs, i.e. the pyrotechnic devices for the airbags, act as actuators, while the driver is a firing unit enhanced with protection and diagnosis features. The power supply unit provides energy to the functional parts, enough for them to act in the event of a collision, and can undertake additional supervision functions.

These functional blocks form the squib driver IC. A standard airbag ECU contains 8 or more firing units, and the number is expected to increase in the future [SB04b]. The system to be validated against requirements is such a firing unit, visible in Figure 5.22.

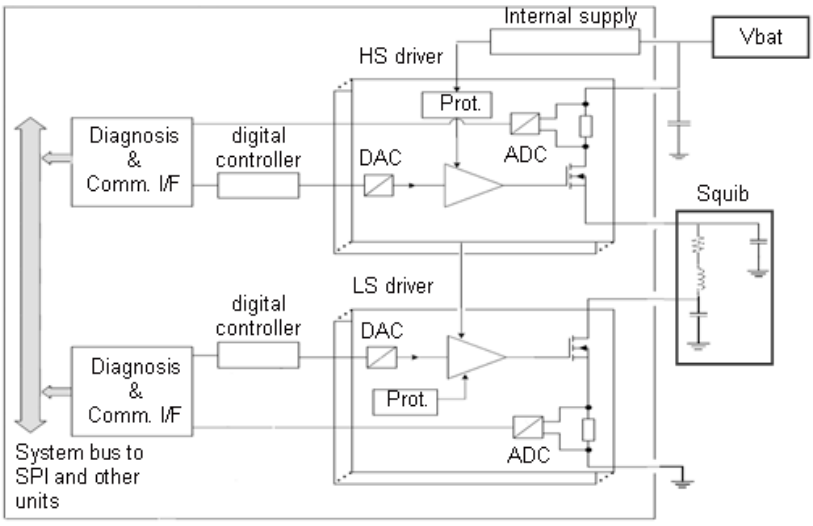


Figure 5.22: The firing unit of an airbag system

At the time of airbag deployment, it must send current of a predefined value through the squib for a few milliseconds, heating it up and thus bringing it to the point of "explosion". To provide the current required for deployment, each firing channel contains one high-side switch (HS) path and one low-side switch (LS) path. The squib is connected between the HS and the LS. Each loop digitally regulates the firing current and contains standard parts of a control loop (analog-to-digital converter (ADC), digital-to-analog converter (DAC), PID controller).

The communication with a microcontroller is realized via an SPI interface. Diagnostic and protection features of the drivers prevent system malfunctions. Several hardware inputs together

with SPI commands control the status of the drivers, to prevent or enable deployment as well as provide diagnostic functions. During normal deployment, the HS operates as a current limiting element, while the LS driver is a current switch with emergency limitation, to protect from firing to short circuit.

5.3.2 Responses and factors

The main test case of interest is the squib deployment. Figure 5.23 shows the nominal simulation results, for the deployment test.

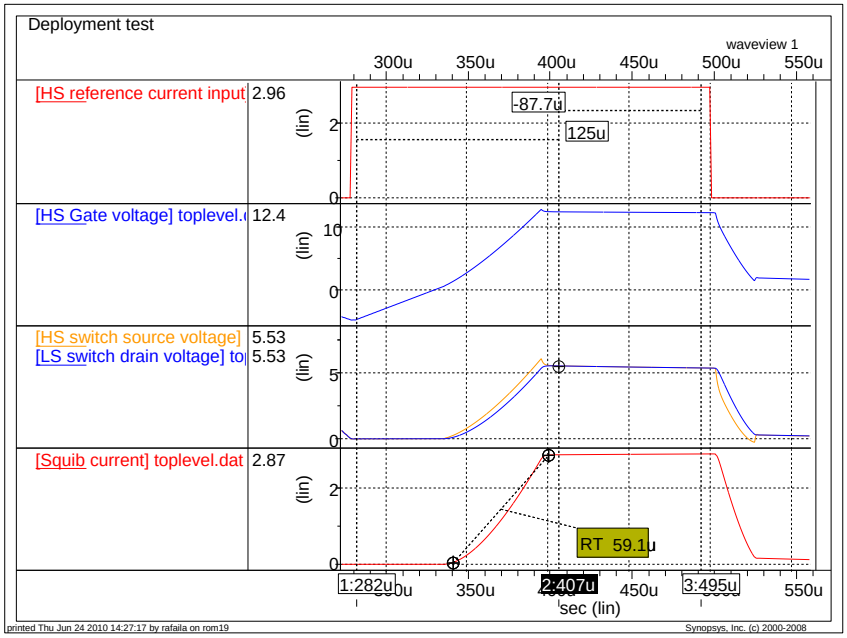


Figure 5.23: Nominal simulation of the deployment test

The SPI command to deploy translates into a stimuli for the reference current of the HS switch, i.e. the HS reference current input. The digital controller drives up the gate voltage for the HS switch, which opens and injects current in the squib. The squib current has a slew rate of 50 milliamperes/microsecond, corresponding to the rise time of $\approx$ 60 microseconds visible on the plot. The final value of approx. 2.9 Amperes is digitally regulated and maintained for about 100 microseconds, when the reference changes back to 0. The waveforms also show the voltages at the two pins connected to the squib terminals.

Each operating condition and functional block plays a role in the firing process, thus has potential impact on the responses: the squib, as load to the HS and LS; functional parameters along the regulation paths; the supply voltage $Vbat$; the SPI interface. These sources introduce the specific factors which are summarized in Table 5.8. The main signal of interest is the current sent through the squib at deployment time. Signal characteristics are chosen as experimental responses: the delay in deployment, slew rate and final value of the current sent to the squib. These and the factors, as extracted from the specification, i.e. with associated ranges and properties, are transferred to the DUT configuration file (Figure 5.24). Since all responses relate to the deployment test, the initial set of factors is common to all responses.

Table 5.8: Factors with impact during deployment

Source of variation		*Factors*
Squib load		Lsquib, Rsquib, Csfx, Csrx
Supply unit		Vbat
HS regulation path	DMOS	HS Vth, HS beta, HS Cgs
	Digital controller	Ctrl. gain, Ctrl. prec., Discretizer block: prec. Df1, offset Df2, threshold Df3
	ADC, DAC	ADC gain, ADC precision, DAC gain
	Other factors	HS Rshunt
LS regulation path	DMOS	LS Vth, LS beta, LS Cgs
	Other factors	LS Rshunt
SPI I/F		SPI Td (Transfer delay)

factor	symbol	description	nominal (=	tolerance	path in model
	Lsquib	Squib load inductance [uH]	...	0.01	dut->i_squib_load->i_l_squib.m_p.value
	Rsquib	Squib load resistance [ohm]	...	...	dut->i_squib_load->i_r_squib.m_p.value
	Csrx	Squib load capacitance (SF to GND)[nF]	...	...	dut->i_squib_load->i_c_sfx.m_p.value
	Csfx	Squib load capacitance (SR to GND)[nF]	...	...	dut->i_squib_load->i_c_srx.m_p.value
	Vbat	Supply voltage [V]	...	...	
	HS beta	HS DMOS threshold voltage [V]	...	...	dut->i_hs_loop-> i_ifx_dmos.m_p.vth
	HS Cgs	HS DMOS beta	...	...	dut->i_hs_loop-> i_ifx_dmos.m_p.beta
	HS Vth	HS DMOS gate-source capacitance [pF]	...	...	dut->i_hs_loop-> i_ifx_dmos.m_p.cgs
	Ctrl gain	Digital controller gain	...	...	dut->i_hs_loop->i_controller.p.g
	ADC gain	Digital controller precision	...	...	dut->i_hs_loop->i_fcn3_1.p.bit_w
	Df1	Discretizer block ADC precision	...	...	dut->i_hs_loop->i_controller->i_discretizer->i_nsh_fcn.p.ad_bit_
	Df2	Discretizer block DAC precision	...	...	dut->i_hs_loop->i_controller->i_discretizer->i_nsh_fcn.p.da_bit_
	Df3	Discretizer block threshold	...	...	dut->i_hs_loop->i_controller->i_discretizer->i_sat_sdf8.p.max_v
	ADC gain	ADC gain	...	...	dut->i_hs_loop->i_controller->i_fcn4.p.ad_g
	ADC prec	ADC precision	...	...	dut->i_hs_loop->i_muls_sdf1.p.scalar
	DAC gain	DAC gain	...	...	dut->i_hs_loop->i_dac_current->i_muls_sdf1.p.scalar
	HS Rshunt	High Side sw. shunt resistance [ohm]	...	...	dut->i_hs_loop->i_rl.m_p.value
	LS Vth	LS DMOS threshold voltage [V]	...	...	dut->i_ls_loop-> i_ifx_dmos.m_p.vth
	LS beta	LS DMOS beta	...	...	dut->i_ls_loop-> i_ifx_dmos.m_p.beta
	LS Cgs	LS DMOS gate-source capacitance [pF]	...	...	dut->i_ls_loop-> i_ifx_dmos.m_p.cgs
	LS Rshunt	Low Side sw. shunt resistance [ohm]	...	...	dut->i_ls_loop->i_r_shunt.m_p.value
	SPI Td	SPI interface transfer delay [us]	...	...	dut->i_spi_adapter.p.delay
response	**name**	**description**	**expected t**	**expected t**	**monitor function call**
	depl_delay	Delay (rising edge of SPI CSQ bit to 10% of current final value) [us]	...	...	meas_delay(dut->i_hs_loop.lim_crt_i, dut_p->current, dut->current_sf_sdf, dut_p->current*0.9)
	depl_slew_rate	Current slew rate [mA/us]	...	...	(dut_p->current*0.8*1.0e3) / meas_delay(dut->current_sf_sdf, dut_p->current*0.1, dut->current_sf_sdf, dut_p->current*0.9)
	depl_current	Settle value of current [A]	...	...	meas_max(dut->current_sf_sdf)

Figure 5.24: DUT configuration file for the airbag system

5.3.3 Results of sequential DoE

Sequential experimentation is performed on the complete set of factors, which sums up to 22. A screening step is first implemented, to reduce the factor set. Then, fixed-level factor DoEs are applied for sensitivity analysis.

Results of factor screening

To filter the important factors a screening phase is necessary. A 32-run R3FF DoE is performed on the complete set of factors. This experiment, detailed in Section 3.2.4.1 is sufficient for screening of all the deployment-related responses. The results are analyzed using ANOVA, to extract the significant factors, as introduced in Section 3.3.1. Table 5.9 comprises the relevant information:

sum of squares SS and probability p of null hypothesis of equality between factor treatments, for each factor. Factors which are kept are filtered on the criteria: $p(f_i) < 0.05$. Checking for a minimum variability of $SS(f_i) > \epsilon$ is also recommended. Values of compliant factors are marked as bold.

Table 5.9: Factor screening results (ANOVA Table)

Response	*Depl. delay*		*Depl. slew rate*		*Depl. current*	
Factor	*SS*	*p*	*SS*	*p*	*SS*	*p*
Lsquib	**1604.7**	**0**	**1.693**	**0**	**4.50e-04**	**0.03**
Rsquib	0.19	0.495	0.046	0.02	0	0.92
Csrx	**74.271**	**0**	0.007	0.29	2.00e-05	0.6
Csfx	**45.551**	**0**	0.002	0.62	3.00e-05	0.52
Vbat	**338.7**	**0**	**0.108**	**0**	1.40e-04	0.17
HS beta	0.055	0.712	**0.925**	**0**	1.00e-05	0.65
HS Cgs	**60.795**	**0**	**53.35**	**0**	1.50e-04	0.16
HS Vth	1.165	0.112	0.004	0.46	1.10e-04	0.21
DAC gain	**75.58**	**0**	**28.26**	**0**	4.00e-05	0.45
ADC gain	0.049	0.727	0	0.86	**0.1082**	**0**
ADC prec	0.009	0.88	0	0.94	**0.102**	**0**
Ctrl gain	0.055	0.712	0.001	0.64	**0.1051**	**0**
Ctrl prec	0.36	0.353	0.001	0.76	**0.1123**	**0**
Df1	0.033	0.772	0.004	0.44	5.00e-05	0.38
Df2	0.684	0.21	0	0.78	1.20e-04	0.2
Df3	2.202	0.038	0.016	0.13	1.00e-05	0.74
HS Rshunt	0.148	0.545	0.012	0.19	3.00e-05	0.49
LS beta	1.007	0.136	0.004	0.44	1.00e-05	0.74
LS Cgs	2.714	0.025	**0.195**	**0**	0	0.8
LS Vth	**41.394**	**0**	0.009	0.26	9.00e-05	0.25
LS Rshunt	0.104	0.612	0.076	0.01	2.00e-05	0.57
SPI Td	**6.604**	**0.002**	0.017	0.13	4.00e-05	0.43
ϵ	3.378		0.054		5.60e-04	

Results of sensitivity analysis

For each response, the sensitivity analysis continues with a R5FF DoE on the remaining factors and as many additional axial runs as necessary, until the residual analysis is satisfactory.

Metamodel validation Figure 5.25 shows plots of residuals after the regression analysis on the results of the CCD DoE, for the deployment delay response. The normal probability plot indicates that they pass the hypothesis of normal distribution, which is a sign of metamodel fitness as explained in Section 3.3.2.2. The scatter plot of residuals against response values indicates no pattern, so there are no systematic prediction errors. This is quantified by a correlation coefficient smaller than the threshold of 0.15, and the maximum absolute residual is smaller than 5%. Therefore, the metamodel passes the validation test.

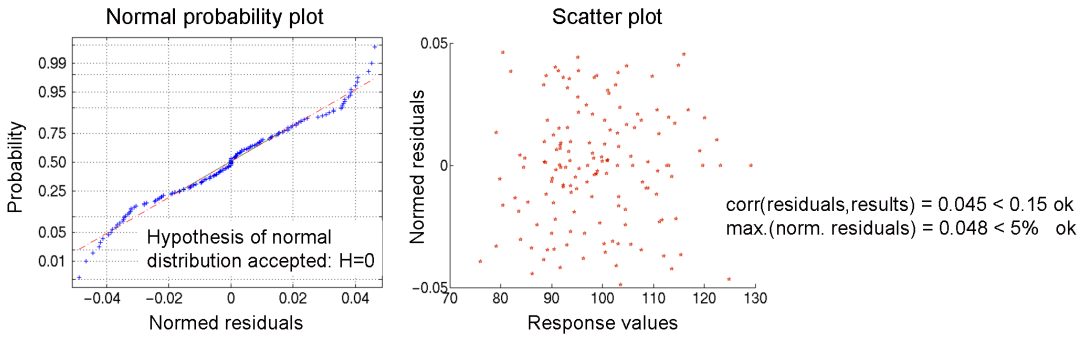


Figure 5.25: Plots of the normed residuals for the deployment delay response

Analysis of factor effects After the residual analysis is satisfactory, the metamodel can be used to extract and interpret factor effects. In order to view individual effects, the metamodel is plotted against each factor, at fixed values of the others. Interaction plots represent the response versus a factor, at various levels of other factors. Figure 5.26 shows individual (a) and 2-factor interaction effects (b).

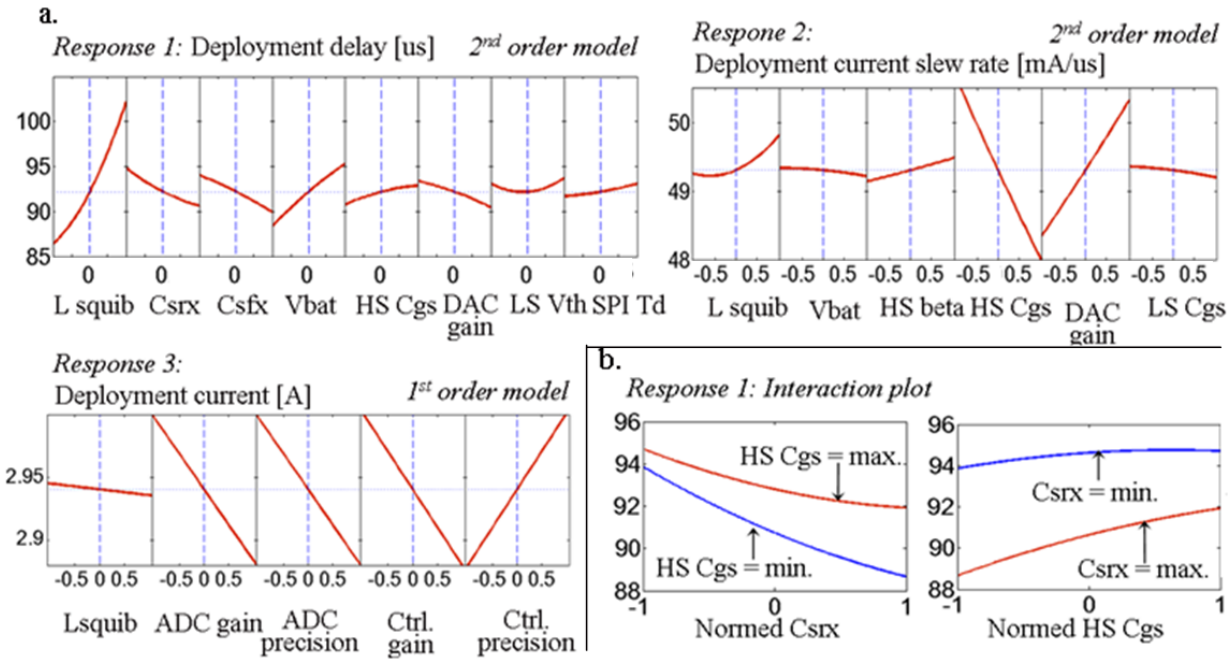


Figure 5.26: Plots of effects on the deployment responses
a. Main effect plots b. Interaction effect plot

Several factors, of electrical nature or functional parameters along the regulation path, impact the responses. The lack of parallelism between curves from the interaction plot indicates 2-factor interactions. The study shows that regression models are reasonable approximations for the responses under study: either a 1^{st} or a 2^{nd} order model with 2-factor interactions is satisfactory. Such an effect analysis ensures a reliable validation, by isolating sources of potential system nonconformance, without the need to have detailed knowledge of block implementation.

5.3.4 Fitting the response distribution after a random factor DoE

As explained in Section 3.4.4.2, the distributions for the three responses are fitted. First, a 500-run experiment on a 6-factor set of randomized factors is performed. Figure 5.27 shows histograms for the distributed factors and scatter plots for the responses' values. They depend not only on the factors' values, but also on their distribution.

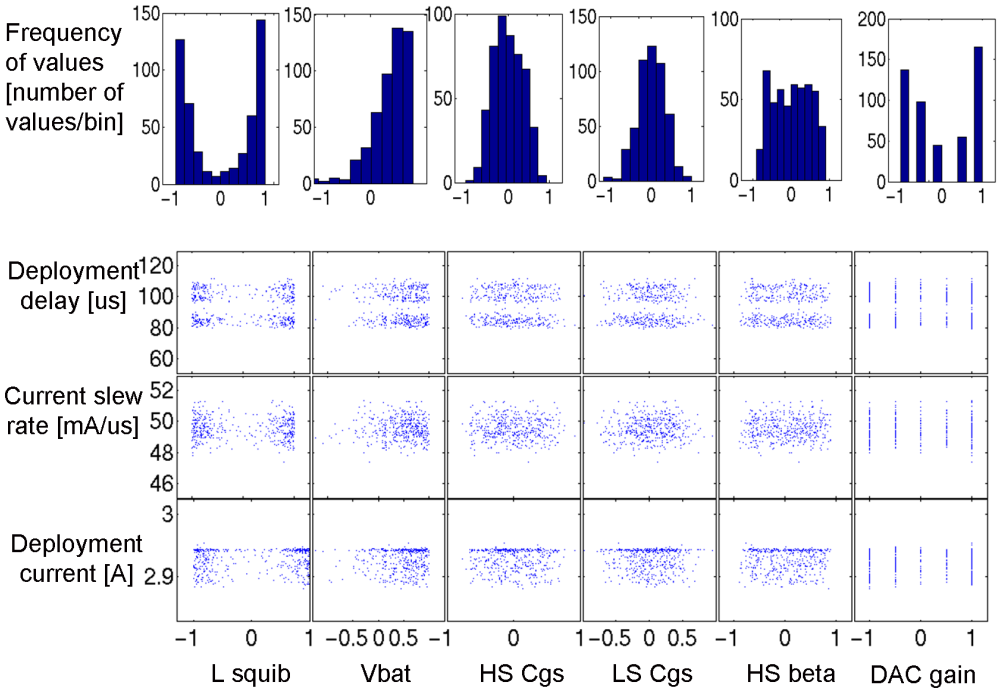


Figure 5.27: Matrix

Figures 5.28, 5.29 and 5.30 plot the simulation results clustered as histograms, and their fitted PDFs. The residual probability values are also marked on the figures. Nonparametric distributions are fitted, since they output better residuals.

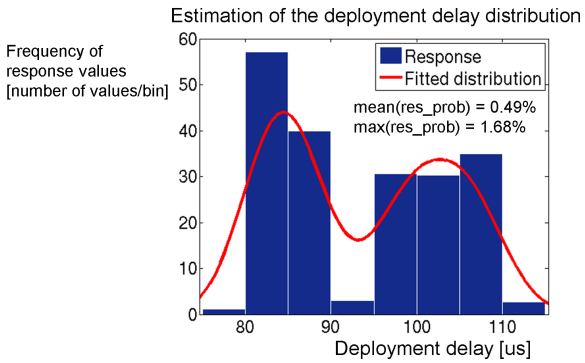


Figure 5.28: Fitting of a nonparametric distribution on response 1

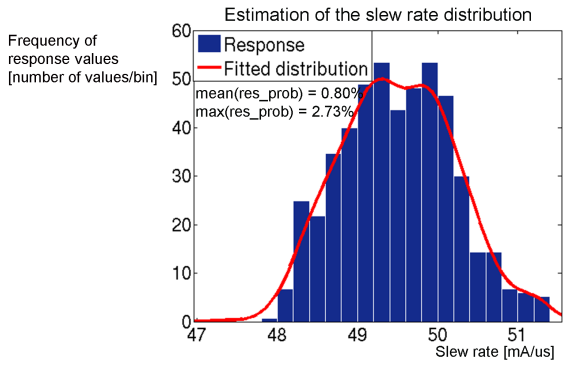


Figure 5.29: Fitting of a nonparametric distribution on response 2

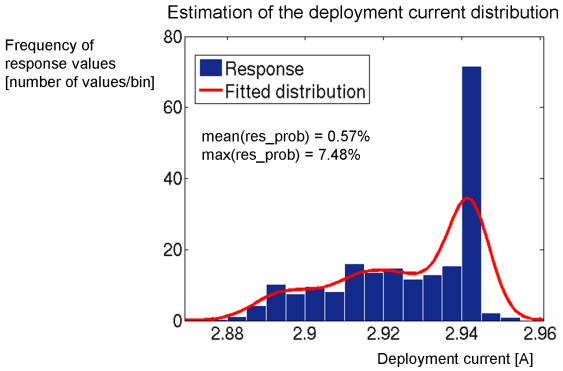


Figure 5.30: Fitting of a nonparametric distribution on response 3

It can be concluded that, as long as the statistical properties of factors are accurately represented, statistical properties of the responses can be extracted. This is of interest to be able to predict the frequency of occurrence for specific values of the response, and estimate the worst-case. Table 5.10 includes these estimates. Mapping the parameters from the factor distributions to the response is difficult, because factors have various distributions, and especially because the response distribution can be nonparametric. Therefore, the sensitivity analysis of RFDoEs stops at estimating the correlations between the response and factor set values.

5.3.5 Worst-case analysis

Starting from the outputs of the previous steps, the worst-case responses are estimated, first as extreme values of the metamodels. Then, the worst-case optimization by a gradient search is applied. Finally, an estimate of the worst-case is extracted from the fitted response PDF.

Worst-cases as metamodel extremes First, the bounds of the regression metamodels are estimated, with respect to the remaining factor set. The method described in Section 3.4.3 is applied. To overcome problems of low convergence because of the interaction effects, the response is evaluated on a grid to reduce the search area. Results are summarized in Table 5.10.

Further iterative worst-case search The worst-case estimation is optimized by gradient-based search. The iterative procedure is conducted in a reduced neighborhood of the previously estimated worst-case, for both response extremes. The search stops when one of the criteria is met, which were detailed in Section 4.2. The results are included in Table 5.10.

Worst-case as extreme of the fitted distribution The worst-case estimates can also be computed as introduced in Section 3.4.3. The distributions fitted as in the previous section are extrapolated to reachable minimum and maximum response values, using a probability threshold of $\epsilon = 1.0e - 6$.

The results for each of the proposed methods are summarized in Table 5.10. The iterative search shows a constant improvement from the staring point, i.e. from the metamodel extreme. The cases with the same worst-case correspond to the worst-case placed in a corner. Since the prediction of the fitted PDF cannot be confirmed, the results of the iterative search are taken as reference, and the offset of the estimated distribution are errors given by its lack of fit: an overall error of 5% relative to the $(max\_response - min\_response)$ value occurs.

Table 5.10: Worst-case found

Response	*Deployment delay [us]*		*Slew rate of current [mA/us]*		*Value of current [A]*	
As metamodel extreme	min 77.679	max 113.000	min 46.607	max 52.349	min 2.700	max 3.185
After iterative search	min 77.679	max 116.714	min 46.489	max 52.424	min 2.695	max 3.186
Estimate of fitted PDF	min 76.103	max 116.140	min 47.431	max 52.054	min 2.702	max 3.191

5.3.6 Alternatives and comparative analysis of performance

For comparison with the proposed approach, three alternatives are implemented:

- Exhaustive corner-case testing DoE: all corner-cases. The initial factor set would require too many runs, i.e. 2^{22} corners, so this test is run only for the subset of factors filtered after screening.
- A Monte-Carlo experiment, with 500 runs with uniformly distributed factors.
- Worst-case direct search, i.e. applied directly on the initial factor set, as explained in Section 5.1.2.

Table 5.11 shows the number of runs invested in each step of the proposed sequential DoE flow. The last line of the table, for the iterative worst-case search, includes the ratio (runs until each extreme response value is reached/runs until the search stops).

The alternatives need more runs: the Monte-Carlo experiment has 500 runs. The worst-case direct search is stopped after maximum $10 \cdot \#factor = 220$ runs, and needs 50 to 200 runs to get to a stop criteria. The Full Factorial has $2^{\#screened\ factors}$ runs, in addition to the 22 initial

Table 5.11: Number of runs in steps of the proposed approach

Step	*Deployment delay*		*Slew rate of current*		*Value of current*	
Center point	**1**					
Screening	**32**					
Regression	8 factors 2^{nd} order model 64 + 2·8 = 80		6 factors 2^{nd} order model 32 + 2·6 = 44		5 factors 1^{st} order model 16	
Iterative search	min 1/18	max 4/24	min 4/18	max 7/18	min 2/15	max 13/15

screening runs. The exhaustive corner-case testing would need more than $4M$ runs, which is not feasible.

The alternatives do not perform sensitivity analysis. Some effects could be visualized only on plots of the Monte-Carlo results, but such an analysis is subjective and not reliable. The effects and interactions are hard to separate and interpret, e.g. as seen in the scatter plots of the deployment responses, in Figure 5.27. Therefore, the proposed approach is more efficient in terms of estimating factor effects.

The proposed approach of iterative search in the reduced area finds better values both for the minimum and maximum response, thus it finds a higher response variability Δ. Table 5.12 shows this relative improvement for each response, computed as the ratio $\delta = (\Delta_{proposed} - \Delta_{alternative})/\Delta_{alternative}$, where $\Delta = (max.\ resp. - min.\ resp.)/mean\ resp.$ As observed in the table, not only does the direct search need more runs, but, when convergent, it finds a poorer response bound. The exhaustive corner-case test also has a lower worst-case performance (can only find the worst corner-case) and it can only be applied after the screening phase. The basic Monte-Carlo is the poorest in terms of worst-case found.

Table 5.12: Relative improvement in the response variability

Alternative	*Delay[us]*			*Slew rate[mA/us]*			*Current[A]*		
	min.	max.	δ	min.	max.	δ	min.	max.	δ
Proposed	77.679	116.714	-	46.489	52.424	-	2.695	3.186	-
Monte-Carlo	80.500	112.214	22.02%	47.270	51.050	56.07%	2.754	3.138	28.10%
Direct search	78.536	115.643	5.08%	46.607	52.275	4.68%	2.717	3.185	5.29%
Full Factorial	77.679	113.000	8.40%	46.786	52.349	6.93%	2.700	3.181	2.08%

6 Discussion

In this chapter, a discussion of the results from the proposed and alternative approaches is presented. Then, a summary of how the initial problems were solved is detailed. Finally, the aspects limiting the approach are identified.

6.1 Comparative analyses

The proposed experiments show an improvement in terms of number of runs, capability of response characterization and worst-case prediction. These were demonstrated in comparison to alternatives both on the window lifter e.g. in Section 5.2.4, Table 5.3, Section 5.2.6, and on the airbag system, e.g. in Section 5.3.6. Performance aspects specific to each method are discussed and concluded next.

Directed test methods versus DoE

Directed test requires the least implementation effort, but is the lowest in terms of performance: the coverage is too low or the effort is too high. Sensitivity analysis is barely addressed, while the worst simulated case is chosen as an estimate for the worst-case.

Trial-and-error The nominal value simulation and best guesses, although effective because of the low number of runs, are highly application dependent, require knowledge of the system and are subjective, thus are not reliable. When the number of factors is high e.g. 10 and interactions or quadratic effects exist, the worst simulated case is a very poor worst-case estimate. The one-at-a-time approach has the main disadvantage of not detecting interaction effects. The custom DoE extends such an approach to sweeping one factor and clustering the others, thus realizes concomitant variations to find interaction effects. Then, a postprocessing step extracts estimates for effects. Even so, the general response variability and worst-case, as presented in Section 5.2.4, are found poorer than DoE-based approaches. Therefore, basic trial-and-error methods are generally low on performance.

Exhaustive search The full set of corners, i.e. the only feasible exhaustive search, presents an exponential increase with the number of factors and easily ends up into more than $1k$ runs. In addition, it assumes that the worst-case is a corner, i.e. that no quadratic effects exist. On the other hand, a R5FF DoE uses much fewer runs (Figure 3.15) to build a 1^{st} order metamodel even with 2-factor interactions, and find its worst corner. This conclusion is supported by results on the window lifter: Table 5.3 shows how easily the worst corner can be detected. When quadratic effects exist, the sequential DoE outputs better results than the exhaustive search, both for the minimum and maximum response. This is shown on the the airbag system, in Table 5.12.

DoE and sequential DoE

Screening experiments can reduce factor sets which would be otherwise difficult to manage. Tables 5.4 and 5.9 show how the ANOVA method, applied after screening experiments, can safely identify insignificant factors out of a larger set. The R3FF DoE easily finds the worst corner-case when the response can be approximated by a 1^{st} order model with no interactions, as best seen in the window lifter results, in Table 5.3. The R5FF DoE can detect interactions, while the CCD DoE is able to find quadratic effects, as observed in the experimental results presented in Figures 5.10, 5.14 and 5.26.

The sequential DoE flow is always indicated, to approach complex responses or large factor sets. A sign of inadequacy simply means the response needs to be approximated by a more complex metamodel. Follow-up DoEs can realize a step-by-step process of metamodel refinement. This is demonstrated on the window lifter system (Section 5.2.6) and on the airbag driver (Section 5.3.3).

Worst-case gradient-based search (direct/in a reduced area) versus genetic algorithm

Section 5.3.5 showed results of the search as a final step after the sequential DoE, while Section 5.3.6 compared it to the direct search, i.e. applied from start on the initial factor set. The direct search needed more runs, as seen in Table 5.11, and it found poorer response bounds, which is visible in Table 5.12. The algorithm does not converge when the factor set and the search area are large, because the underlying assumptions do not hold for the complete space.

The genetic algorithm approach (GA) generates subsequent populations of solutions, consisting of several samples each, which must be simulated. Results were compared against gradient-based methods on the window lifter system, in Section 5.2.6. These results show that the use of random number generators to apply changes from a generation to the next do not help to approach the real worst-case, especially for large factor sets. The consequence is a lower performance of the GA method, both in terms of worst-case and of number of runs. Therefore, the GA strategy should be applied to solve worst-case problems on responses which are either discontinuous, nondifferentiable, stochastic or of high orders.

Monte-Carlo versus RFDoEs versus fixed-level DoEs

A lower performance of the Monte-Carlo experiments as compared to other methods is observed in the results. A smaller response variability out of a higher number of runs is recorded in the case of the window lifter system, in the comparison Section 5.2.4 and in Table 5.6, as well as for the airbag system, in Table 5.12. Any sensitivity data extracted from pure Monte-Carlo simulations is graphical, thus subjective and not reliable, as effects are hard to separate and interpret, as seen

in Figure 5.7. CCD DoE can yield outputs comparable to Monte-Carlo out of much less data, as pointed out in Figure 5.10.

The basic differences between Monte-Carlo and RFDoEs were explained in Section 3.2.5. From a planning point of view, they involve comparable effort. Both Monte-Carlo and RFDoEs help to cover the verification space better and observe how statistical factor properties impact the response, as demonstrated on the window lifter (Section 5.2.3) and on the airbag system (Section 5.3.4). RFDoE methods, on the other hand, are more expressive in terms of statistical properties i.e. cover a wide range of functions and can express correlations between factors. Then, they extract correlations factors-responses, to measure and compare factor effects.

The response characterization as a random variable is another advantage of RFDoEs. By approximating the response distribution, a probability is associated to each response value. Such estimations were demonstrated both for the window lifter (Figure 5.8) and for the airbag system (Figures 5.28, 5.29, 5.30). The worst-case response is estimated as the extreme of the fitted distribution, reached with a probability higher than a given threshold. This is exemplified in Section 5.3.5, in Table 5.10, with worst-case estimates close enough to the confirmed results of the sequential DoE.

Alternatively, regression can be performed on the RFDoE results, and the analysis can continue as for fixed-level factor DoEs. Figure 5.9 points out that regression on RFDoEs can discover quadratic effects and interactions. Moreover, regression on the results of a considerable number of runs confirms the adequacy of polynomial metamodels as general representation for major effects in such systems. Therefore, sensitivity analysis by a study of correlation or, alternatively, a metamodelling step, followed by worst-case analysis of the response distribution or, respectively, of the fitted metamodel, yield experiments much superior to the basic Monte-Carlo.

Comparative analysis

A comparison of the methods, according to the criteria included in Table 3.4, is summarized in Table 6.1 (a + stands for higher performance). When time efficiency is the main target, sequential DoE is considered the best choice. The characterization of individual and interaction factor effects and the predictive nature of the metamodel are the main benefits. The iterative search does not improve the sensitivity analysis, but can get closer to the real worst-case. RFDoE invest more runs, but reach a better coverage and can extract statistical measures for factor effects and response properties.

6.2 Summary

The next paragraphs detail how the initial targets set in Section 1.3 and evaluated in Table 6.1 were achieved, and their related advantages.

Sensitivity analysis: Response characterization with respect to the factors is performed by extracting a set of coefficients, at least one per factor. This is necessary to support the verification engineer to identify the sources of potential failure of the system. The effects of factors and interactions on the system behaviour can be correctly interpreted. None of the alternatives from Table 6.1 realizes such a sensitivity analysis.

Method	*No. sim. runs*	*Worst-case response*	*Effects estimates*	*Graphical representation*	*Statistical factor properties*	*Implementation effort*
Proposed approach						
Sequential DoE	++	++	++	+++	-	+
Iterative search	++	+	-	-	-	+
RFDoE	+	+	+	++	++	++
Alternative approach						
Exhaustive search (full factorial)	-	-	-	+	-	++
Monte-Carlo	+	-	-	+	+	++
Genetic Algorithm	-	-	-	-	-	+

Table 6.1: Comparative analysis

Worst-case analysis: The worst-case can be estimated as the metamodel extreme. It is confirmed by a check of metamodel adequacy and by an additional iterative gradient-based search in the neighborhood of the prediction. Alternatively, RFDoEs can predict the worst-case as the extreme reached by the estimated PDF, extracted from the response distribution. It cannot, however, be confirmed by simulation. As compared to the alternatives, the proposed worst-case analyses output better results.

Transient analysis: The problem amplifies when the response varies in time, e.g. when it is a signal. The effects of variations and the conditions which drive the system into a worst-case also change in time. In order to validate the behaviour of signals of interest, for the duration of the complete test, the approach is extended for a transient response analysis. Metamodels are fitted in equidistant samples of the time frame. The factor effects, as well as metamodel bounds and corresponding factor sets are estimated and validated. Then, margins for the analyzed samples are assembled, to build safe bounds for the signal. The same number of runs as in the case of a static response is involved. Each step is monitored closely, in order to identify and control potential bottlenecks of simulation or processing. The trade-off between the overall time versus the complexity which is handled recommends such extended DoE flows for future use.

Number of runs: A reduced number of simulation runs is sufficient to find effects and to predict the behaviour between the simulated points. The number of runs does not increase exponentially with the number of factors, as happens with exhaustive search methods, and is smaller than $1k$ runs, even for factor sets up to 30. Based on the sparsity of effects principle, several factors are insignificant and can be screened out. Minimum sized experimental designs allow screening and estimation of effects for the remaining factors. Alternatively, RFDoEs invest more runs, but the number is controlled to reach the desired coverage. Selected alternatives, i.e. basic random test, directed test or GA methods, involve more runs.

Graphical analyses Graphical analyses are an important part for a proper interpretation of the effects and discovered worst-cases. Plotting the response metamodel determines a better understanding of the factor effects. The interaction plots in particular reveal effects which can hardly be tracked or viewed otherwise, and which should not be omitted in a rigorous analysis. Residuals should also be visualized to make sure there are no systematic prediction errors of the metamodel. With RFDoEs, the factors' and response distributions, as well as correlated sets of values should be plotted. In the case of transient response analysis, the graphical analysis of predicted signal bounds, confirmation simulations, as well as factor effects which change in time is important.

Statistical properties By accounting for the real distribution of factors and correlations between them, a real distribution for the response is simulated and characterized. Correlations between the data sets response-factors reveal another side of the sensitivity analysis problem, while the worst response case can be estimated by fitting the empirical response PDF.

Implementation effort As intended from the start, the portability and applicability of the methods was maximized. By decoupling the pre-, postprocessing and the simulation control algorithms, any implementation effort is compensated by a high reusability. The DUT model setup and configuration require minimum effort and are kept separate from the experiment planning and analysis, so that the flow is as little as possible application dependent. This is certainly not the case with the alternative best-guess methods. GA needs not only the initial implementation effort, but also a tuning of the algorithm parameters depending on the number of factors and the complexity of the response.

6.3 Limitations

The presented approach has restrictions reflected in the size of the verification space. The complexity increases with the number of factors, so it is recommended to keep the initial set to no more than 50 factors, since that would introduce scaling problems, i.e. too many required runs, difficulty to design proper experiments. No more than 20 factors should present significant effects, otherwise for the regression phase it is difficult to estimate the high number of interaction coefficients, i.e. $n \cdot (n-1)/2$.

Another limitation is given by assumptions on the response. One of the proposed metamodels should fit, and this can be assessed by a proper validation phase, preferably in a complete sequential DoE flow. In case even after all sequential steps and metamodel optimizations, a lack of fit still exists, the RFDoEs can always be counted on. They involve less planning and analysis effort, but for reliable conclusions and an acceptable coverage, they must invest sufficient runs. A second option when no metamodel fits is to reduce the number of factors which are initially taken into account, or at least analyze them separately. Their ranges can also be reduced, because complex factor interaction and individual effects are more likely to occur when factor sets and ranges are larger.

The factors' ranges should be limited, e.g. variations up to 50% relative to the nominal value. This is to reduce the risks of metamodel unfitness, because discontinuities in the response and even misbehaviours of the system can occur when wider ranges must be covered.

7 Conclusion and outlook

This chapter includes a final argumentation of the contributions and benefits of the present work. Then, several possible directions to continue are presented.

7.1 Conclusion

The industry of automotive electronics must cope with great challenges because their complexity increases faster than the current verification flows can handle. On the other hand, pre-silicon verification must provide as fast as possible a highly reliable outcome, especially in the automotive sector. Ensuring safety is essential in this field, because any design problem which is not detected in good time can lead to system misbehaviour, and finally to product re-spins or even threaten the lives of the end users. Hence the need to verify starting from early development phases that the reliability is maximum i.e. that the probability of system failure is minimum.

Since multiple sources of variation influence the system behaviour, multivariate sensitivity analysis are needed to find the effects of variations on the system responses. Then, a search for the worst response case is a must, in order to estimate where the system could fail the conformance against requirements. A reduced number of simulations must be invested in this process. The current industrial practices address these issues too late, i.e. at a low level of abstraction e.g. circuit-level. Thus they can either handle a lower complexity than is needed i.e. a small number of variations, or they require more time than can be spent i.e. too many simulation runs.

The approach proposes a flow for simulation-based experimentation on complex systems. First, a framework which couples the simulator with algorithms for control and postprocessing of multi-simulation experiments is built. The framework allows transfer of the specification requirements as algorithm inputs. Principles and methods for experiment planning and data analysis are adopted from the field of statistical DoE. Their feasibility is tested on the systems to be validated and modifications are introduced where needed. Classical and custom experimental designs, metamodels of the results space and fitting algorithms are evaluated and a selection of effective methods is made. Such DoE and metamodelling strategies, as well as random test methods, are adopted and optimized.

This way, effects of variations and the worst response case are estimated. To achieve an optimal trade-off between the accuracy of estimates and the number of invested runs, a sequential experimentation flow is proposed, which includes:

- factor screening, to reduce the size of the verification space
- sequential DoE for sensitivity analysis and to build an adequate metamodel
- estimation of the worst-case and iterative search to increase its reliability

To approach a better coverage in terms of statistical properties of factors and the response, random test methods are extended to a new type of experimental design, to perform:

- factor randomization, accounting for their statistical properties and correlations
- extraction of factor effects as correlations between the sets of factors-response values
- characterization of the response distribution and estimation of its extremes

The issues are addressed more effectively than with existing trial-and-error methods or more complex gradient-based or genetic algorithms. The experimentation steps are extended by optimization of the worst-case search and automation to form a sequential self-correcting flow. Transient response analysis is addressed last, by extending the basic flow to analyze time variant responses, for which the factors' effects and the worst-case conditions also vary in time.

The proposed methods are demonstrated on two automotive case-studies: a window lifter system and an airbag control unit. No output of the simulation experiment is wasted and the postprocessing effort is controlled and small enough. Results show that the statistical methods under evaluation form an efficient solution to cope with variations in complex automotive systems. This way, the gap between research, which extensively supports these methods, and simulation practices in industry can be covered.

7.2 Outlook

Further work can address the topics detailed below.

Handling more factors The main issue to address is the scaling problem which occurs with an initial set of more factors e.g. > 50. A hierarchical approach is possible, i.e. split the initial set and cluster the factors into subsets with more probable interactions. The factors of subsets can be chosen as parameters of the same subsystems or of subsystems which directly interact in the simulated test. The experimentation flow can be applied separately on each subset. Then, a custom analysis of the interaction between clusters could be performed, similarly to the custom DoE approach.

More metamodelling representations It is also of interest to find other metamodels appropriate for common effects in the systems under verification. For instance, response discontinuities can occur e.g. in the form of piece-wise constant response(factor). In such cases, interpolating the response over the factor space is still possible and useful, but care must be taken so that both individual and interaction factor effects to be accounted for in the new metamodel. Other shapes of responses can be fitted, and included into the metamodel optimization flow.

Sequential RFDoE Extending the RFDoEs to change properties in the factor distributions and identify changes in the response distribution is also of interest because it would address the sensitivity analysis problem. Mapping the response distribution to the statistical properties of factors depends, however, on each distribution type, for the factor set and for the response. If this mapping would be established, the factor distributions can be changed in order to drive the response in the worst-case. Although this would depart from keeping the factor distributions real, such an adaptive flow is an interesting direction to improve the worst-case analysis.

Multi-objective DoE This refers to metamodelling a vector of responses, with respect to the same factor set. The analysis must be extended to consider the correlations between responses, and the fact the not all worst responses' cases can be reached simultaneously. In this case, "cost" functions could be attached to responses to prioritize them.

A Listings

A.1 Pseudo-code for the DoE matrix generation

The pseudo-code below generates the DoE matrix. The used routines are:

- `get_resolution` computes the maximum resolution which can be achieved by a fractional factorial which has a number of runs smaller than the input number of runs.
- `generate_ff` generates the corresponding fractional factorial using the MATLAB functions *fracfact* and *factgen* (see Table B.4 in the Appendix).
- `generate_ccd` extends the fractional factorial to a CCD DoE.
- `augment_doe` augments the input DoE matrix with lines, using one of the available algorithms (*lhsdesign*, *cordexch*, *rowexch* MATLAB algorithms). The `'rf'` option uses the structure of statistical properties of the factors (distributions, correlations) to generate random numbers using linear correlation and the supported distributions).

```
function doe = get_doe_matrix(factor_nr, max_run_nr, doe_type, factor_stat_struct)
 if nargin==2 or doe_type=='basic'
   resolution = get_resolution(factor_nr, max_run_nr-2*factor_nr-1)
   ff_doe = generate_ff(factor_nr, resolution)
   left_run_nr = max_run_nr - size(ff_doe(:,1))
   if left_run_nr>2*factor_nr+1
     ccd_doe = generate_ccd(ff_doe)
     left_run_nr = max_run_nr - size(ccd_doe(:,1))
     if left_run_nr>0
       doe = augment_doe(ccd_doe,left_run_nr,'lhs')
     else
       return ccd_doe
     end
   else
     return ff_doe
   end
 else
   doe = zeros(1,factor_nr)
   if nargin>3
```

```
   % factor_stat_struct => statistical properties
     doe = augment_doe(doe,max_run_nr-1,'rf',factor_stat_struct)
   else
     doe = augment_doe(doe,max_run_nr-1,'lhs')
   end
   return doe
 end
end
```

A.2 Pseudo-code for the analysis of results

`do_regression` performs regression on the experiment results. `validate` makes the residual analysis. The MATLAB functions which are used `regress, corr, lillietest, x2fx` are included in the appendix Table B.4. `evaluate(doe_matrix,coef)` returns the vector of values of the polynomial with coefficients `coef` at the values rows of the `doe_matrix`.

```
function [coef,residuals_ok] = do_regression(doe_matrix,r,type)
 % type = linear, quadratic,...

 [run_nr,factor_nr] = size(doe_matrix)

 % default 80% of the total runs used for regression
 % but must be more than the minimum sized basic doe
 regress_run_nr = max(run_nr*0.8,length(get_doe_matrix(factor_nr,run_nr,'basic')'))

 X = x2fx(doe_matrix(1:regress_run_nr,:),type)
 coef = regress(X,r(1:regress_run_nr))

 validate_run_nr = run_nr-regress_run_nr
 residuals_ok = validate(doe_matrix,r(validate_run_nr+1:run_nr),coef)

 return [coef,residuals_ok]
end

function residuals_ok = validate(doe_matrix,r,coef)
 residuals = r - evaluate(doe_matrix,coef)
 norm_res = residuals/(max(r) - min(r))
 if corr(norm_res,r)>0.15
   residuals_ok.corr_ok = 0
 else
   residuals_ok.corr_ok = 1
 end
 if mean(abs(norm_res))>0.01
   residuals_ok.mean_ok = 0
 else
   residuals_ok.mean_ok = 1
```

```
 end
 if max(abs(norm_res))>0.05
   residuals_ok.max_ok = 0
 else
   residuals_ok.max_ok = 1
 end
 residuals_ok.h = lillietest(norm_res)
 return residuals_ok
end
```

A.3 Pseudo-code for the sequential algorithm

The pseudo-code for the sequential experimentation algorithm is inserted on the next page. The notations which are used are:

- n - number of factors.
- r - the vector of response results, $R(i)$ - the vector of metamodel coefficients at iteration i
- doe - sampling function, which creates a hierarchy of designs, $doe(k, f) \subset doe(k+1, f)$, for each factor set f; it is defined below.
- k - model complexity, (an iteration which does not change the factor set), is 0 for a linear model; 1 for a first order model; $k \geq 2$ for a model with 2-factor interactions and individual effects of orders k; $kmax$ can be set between 2 and 5.
- i - iteration which changes the factor set.
- $f(i)$ - vector of active factors for iteration i: a bit is 1 if its factor is active, 0 if not.

The doe function is defined as:
$doe(0, f) = R3FF \cup [0...0]$
$doe(1, f) = R5FF \cup [0...0]$
$doe(2, f) = CCD$
$doe(k, f) = doe(k-1, f) \cup (-1)^k \cdot \alpha^{k-2} \cdot f \cdot Id(n)], k < kmax$, where α is between 0 and 1
When $k \geq kmax$, $doe(k, f) = doe(k-1, f) \cup lhs(f, max\_run\_nr - size(doe(k-1, f)))$, where lhs generates the LHS DoE corresponding to the number of factors and number of runs parameters. Alternatively, $doe(k, f) = doe(k-1, f) \cup ff(r+1, f)$, where $ff(r, f)$ generates the fractional factorial of the next resolution r, to estimate higher-order interaction effects.

The procedures which are used are:

- run: launches simulation runs with factors set to the values of matrix d
- $find\_min$: finds the metamodel minimum value and the corresponding factor set, as explained in Section 3.4.3
- $reduce(f, R)$: resets one bit from the vector f, for the lowest main coefficient of R
- $extend(f, R)$: sets one bit from the vector f, for the highest main coefficient of R
- function $[coef, residuals\_ok] = do_regression(d, r, type)$, from A.2
- function $residuals\_ok = validate(d, r, coef)$ from A.2

The performance will be given by the final residual metrics and by the final $size(doe(k))$.

```
    i = 1
    if n > 10
      k = 0, goto 0
    else
      k = 1, goto 1
0:  d(i) = doe(f(i),k)
    r = run d
    f(1) = anova(d(i),r)
1:  d(i) = doe(f(i),k)
    r = run d(i) \ d(i-1)             % run only what is new
    [R(i),residuals_ok] = do_regression(doe(i),r,k)    % k identifies the R type
    if residuals_ok = false
      if k < kmax
        k = k+1                       % type(a):k<kmax or type(b):k>=kmax
        if sum(size(d(i)))>mmax
          goto 2
        else
          goto 1
      else
        i = i+1
        f(i) = reduce(f(i-1),R(i))    % type(c):k=0
        k = 0
        if sum(size(d(i))) > mmax
          goto 2
        else
          goto 0
    else if sum(f(i)) < n
      i = i+1
      f(i) = extend(f(i-1),R(i-2))    % extend factor set (identify initial error)
      k = 1
      if sum(number_rows(d(i))) > mmax
        goto 2
      else
        goto 1
    else
      goto 2
2:  [min -max] = [find_min(R) find_min(-R)]
    r = run([min max])
    res = (R([min max])-r)/(max(abs(r)) - min (abs(r))
    if abs((res)) > thr_max_res
      f(i) = reduce(f(i-1))
      if sum(size(d(i))) > mmax
        exit
      else
        goto 0                        % start over
    else end                          % successfully reached the end
```

B Tables

B.1 Number of runs in fixed-level DoEs

Table B.1: Number of runs in fixed-level DoEs

Number of factors	*R3FF*	*R4FF*	*R5FF*	*CCD DoE*
1	1	1	1	5
2	4	4	4	9
3	4	8	8	15
4	8	8	16	25
5	8	16	16	27
6	8	16	32	45
7	8	16	64	79
8	16	16	64	81
9	16	32	128	147
10	16	32	128	149
11	16	128	128	151
12	16	256	256	281
13	16	256	256	283
14	16	256	256	285
15	16	256	256	287
16	32	256	256	289
17	32	256	256	291

B.2 Selected fractional factorials

Table B.2 on the next page includes the resolution, number of runs and design generators for selected fractional factorial designs, for a number of factors up to 15 and a number of runs of up to 128.

Table B.2: Selected fractional factorials (source: [Mon05], Table 8-14)

Nr factors	*Fraction*	*Nr runs*	*Design generators*
3	2^{3-1}_{III}	4	C=± AB
4	2^{4-1}_{IV}	8	D=± ABC
5	2^{5-1}_{V}	16	E=± ABCD
	2^{5-2}_{III}	8	D=± AB
			E=± AC
6	2^{6-1}_{IV}	32	F=± ABCDE
	2^{6-2}_{IV}	16	E=± ABC
			F=± BCD
	2^{6-3}_{IV}	8	D=± AB
			F=± BC
7	2^{7-1}_{VII}	64	G=± ABCDEF
	2^{7-2}_{IV}	32	F=± ABCD
			G=± ABDE
	2^{7-3}_{IV}	16	E=± ABC
			F=± BCD
			G=± ACD
	2^{7-4}_{III}	8	D=± AB
			E=± AC
			F=± BC
			G=± ABC
8	2^{8-2}_{V}	64	J=± ABCD
			H=± ABEF
	2^{8-3}_{IV}	32	F=± ABC
			G=± ABD
			H=± BCDE
	2^{8-4}_{IV}	16	E=± BCD
			F=± ACD
			G=± ABC
			H=± ABD
9	2^{9-2}_{VI}	128	H=± ACDFG
			J=± BCEFG
	2^{9-3}_{IV}	64	G=± ABCD
			H=± ACEF
			J=± CDEF
	2^{9-4}_{IV}	32	F=± BCDE
	2^{9-4}_{IV}	32	F=± BCDE
			G=± ACDE
			H=± ABDE
			J=± ABCE
10	2^{10-3}_{V}	128	H=± ABCG
			J=± ACDE
			K=± ACDF
	2^{10-4}_{IV}	64	G=± BCDF
			J=± ABDE
			K=± ABCE
	2^{10-5}_{IV}	32	F=± ABCD
			G=± ABCE
			H=± ABDE
			J=± ACDE
			K=± BCDE
	2^{10-6}_{III}	16	E=± ABC
			F=± BCD
			G=± ACD
			H=± ABD
			H=± ABDE
			K=± AB
11	2^{11-5}_{IV}	64	G=± CDE
			H=± ABCD
			J=± ABF
			K=± BDEF
			L=± ADEF
	2^{11-6}_{IV}	32	F=± ABC
			G=± BCD
			H=± CDE
			J=± ACD
			K=± ADE
			L=± BDE
	2^{11-7}_{III}	16	E=± ABC
			F=± BCD
			F=± BCD
			G=± ACD
			H=± ABD
			J=± ABCD
12	2^{12-8}_{III}	16	E=± ABC
			F=± ABD
			G=± ACD
			H=± BCD
			J=± ABCD
			K=± AB
			L=± AC
			M=± AD
			F=± ABD
			G=± ACD
			H=± BCD
			J=± ABCD
			K=± AB
			L=± AC
			M=± AD
			N=± BC
14	2^{14-10}_{III}	16	E=± ABC
			F=± ABD
			G=± ACD
			H=± BCD
			J=± ABCD
			K=± AB
			L=± AC
			M=± AD
			N=± BC
			O=± BD
15	2^{15-11}_{III}	16	E=± ABC
			F=± ABD
			G=± ACD
			H=± BCD
			J=± ABCD
			K=± AB
			L=± AC
			M=± AD
			N=± BC
			O=± BD
			P=± CD

B.3 Number of runs in Latin Hypercube Sampling DoEs

The number of runs versus distance is estimated using the *lhsdesign* and *pdist* MATLAB functions. The *distance* is the Euclidean distance between pairs of sample points in the factor space.

Table B.3: Number of runs in space filling DoEs

Number of factors	*distance=0.1*	*distance=0.2*	*distance=0.3*	*distance=0.4*	*distance=0.5*
3	33	16	11	7	6
4	85	32	15	13	8
5	159	49	27	21	10
6	757	121	50	37	15
7	1814	211	81	50	19
8	-	500	116	73	27
9	-	900	330	118	37
10	-	-	490	211	66
11	-	-	991	364	166
12	-	-	-	589	181
13	-	-	-	833	352
14	-	-	-	-	561
15	-	-	-	-	1021

B.4 Probability distribution functions

B.4.1 MATLAB probability distribution functions

Specific functions for each supported distribution:

- Parameter estimation functions (fit)
- Cumulative Distribution Functions (cdf)
- Probability density functions (pdf)
- Inverse cumulative distribution functions (inv)
- Random number generator functions (rnd)

Supported distributions and corresponding functions:

- normal (normfit, normcdf, normpdf, norminv, normrnd)
- uniform (unifit, unifcdf, unifpdf, unifinv, unifrnd)
- extreme value (evfit, evcdf, evpdf, evinv, evrnd)
- exponential (expfit, expcdf, exppdf, expinv, exprnd)
- binomial (binofit, binocdf, binopdf, binoinv, binornd)

B.4.2 Custom probability distribution functions

- corner-peaked, minimum-peaked, maximum-peaked (derived from exponential or normal PDFs)
- piece-wise linear cumulative density function (derived from the statistical package: [MHE08])
- piece-wise constant cumulative density function (derived from the statistical package: [MHE08])

B.5 MATLAB functions

Table B.4: MATLAB functions, used in the experiment planning and analysis algorithms

Mnemonic	*Description*
Statistics toolbox	
anovan	N-way analysis of variance (ANOVA)
corr	Linear or rank correlation
fracfact	Generate fractional factorial design from generators.
fracfactgen	Output fractional factorial design generators.
ksdensity	Compute density estimate using kernel-smoothing method.
lhsdesign	Generate latin hypercube sample.
lhsnorm	Generate latin hypercube sample with normal distribution.
lillietest	Test the hypothesis that the input data has a normal distribution, returns 1 if it can be rejected and 0 if not.
mvrnd	Random matrices from multivariate normal distribution.
normplot	Normal probability plot for graphical normality testing.
pdist	Compute pairwise distance between observations.
polyfit	Polynomial curve fitting.
polyval	Polynomial evaluation.
regress	The least squares fit of the vector of observations on the matrix of regressors, returns the vector of coefficient estimates.
regstats	Regression diagnostics for linear model.
robustfit	Robust linear regression to fit the vector of observations as a function of the columns of the matrix of regressors, returns the vector of coefficient estimates.
rstool	Interactive fitting and visualization of response surfaces.
x2fx	Transform matrix of variable values to design matrix of term values (regressor).
Optimization toolbox	
fmincon	Find minimum of constrained nonlinear multivariable function
fminsearch	Find minimum of unconstrained multivariable function using derivative-free method
fminunc	Find minimum of unconstrained multivariable function
linprog	Solve linear programming problems
quadprog	Solve quadratic programming problems
Global optimization toolbox	
ga	Find minimum of function using genetic algorithm using callbacks to the to the objective function, i.e. the simulator.

C Analysis of variance for one factor

The fundamental ANOVA identity for the one factor f case states that:

$$SS_{total} = SS_{treatments} + SS_{error} \tag{C.1}$$

where:

$$SS_{treatments} = n \cdot \sum_{i=1}^{a} (r_{i.} - r_{..})^2 \tag{C.2}$$

$$SS_{error} = \sum_{i=1}^{a} \sum_{j=1}^{n} (r_{ij} - r_{i.})^2 \tag{C.3}$$

$r_{i.} = 1/n \cdot \sum_{j=1}^{n} r_{ij}$ and $r_{..} = 1/a \cdot \sum_{i=1}^{a} r_{i.}$. r_{ij} is the response of run j, from the set of runs where f is set to level i. a is the number of levels for f. n observations are taken, for each level of f. N is the total number of observations. MS is the mean square of sums, defined as the sum of squares divided by the degrees of freedom:

$$MS_{factor} = SS_{factor}/(a-1) \tag{C.4}$$

The fundamental ANOVA identity C.1 states that the total variability in the data, as measured by the total sum of squares, can be partitioned into a sum of squares of the differences between the treatment averages and the grand average, plus a sum of squares of the differences of observations within treatments, from the treatment average.

If the null hypothesis about the equality of treatments with different factor levels is true, then, according to Cochran's theorem, the ratio:
$H_0 :\ F_0 = MS_{treatments}/MS_{error}$ is distributed as F with $a-1$ and $N-a$ degrees of freedom.

H_0 is the test statistic for the hypothesis of no differences in treatment means. Under the alternative hypothesis, the expected value of the numerator $MS_{treatments}$ is greater than the expected value of the denominator MS_{error}. So, the null thesis H_0 is rejected on values of the test statistic that are too large.

It is therefore concluded that there are differences in the treatment means if $F_0 > F_{\alpha,a-1,N-a}$, where F_0 is computed from the test statistic equation. α is the confidence level, typically 0.05 corresponding to a 95% confidence in deciding on the statistic. Alternatively, the p-approach to make the decision computes and uses the probability p intead of the statistic F_0. More details on how to compute the F and p values can be found in [Mon05, p.65-75].

Literature

[ADR+08] Alassir, M. ; Denoulet, J. ; Romain, O. ; Suissa, A. ; Garda, P.: Modelling Field Bus Communications in Mixed-Signal Embedded Systems. In: *EURASIP Journal on Embedded Systems*. Hindawi Publishing Corporation, 2008

[ATW06] Ayeb, M. ; Theuerkauf, H. ; Winsel, C.W.T.: Robust identification of nonlinear dynamic systems using Design of Experiment. In: *IEEE Computer Aided Control System Design*, 2006, p. 2321–2326

[Bar04] Barton, R.R.: Designing simulation experiments. In: *Winter Simulation Conference*, 2004

[FCS08] Fu, Michael C. ; Chen, Chun-Hung ; Shi, Leyuan: Some topics for simulation optimization. In: *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, Winter Simulation Conference, 2008. – ISBN 978–1–4244–2708–6, p. 27–38

[FGA05] Fu, Michael C. ; Glover, Fred W. ; April, Jay: Simulation optimization: a review, new developments, and applications. In: *WSC '05: Proceedings of the 37th conference on Winter simulation*, Winter Simulation Conference, 2005. – ISBN 0–7803–9519–0, p. 83–95

[FS00] Femia, N. ; Spagnuolo, G.: True worst-case circuit tolerance analysis using Genetic Algorithms and Affine Arithmetic. In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 2000

[GBVE08] Grimm, C. ; Barnasconi, M. ; Vachoux, A. ; Einwich, K. *An introduction to modeling Embedded Analog/Mixed-Signal Systems using SystemC AMS extensions.* 2008

[GCLD10] Gorissen, D. ; Couckuyt, I. ; Laermans, E. ; Dhaene, T.: Multiobjective global surrogate modeling, dealing with the 5-percent problem. In: *Engineering With Computers manuscript* (2010), p. 81–98

[GED07] Grosse, Daniel ; Ebendt, Rüdiger ; Drechsler, Rolf: Improvements for constraint solving in the systemc verification library. In: *GLSVLSI '07: Proceedings of the 17th ACM Great Lakes symposium on VLSI*. New York, NY, USA : ACM, 2007. – ISBN 978–1–59593–605–9, p. 493–496

[Ghe05] Ghenassia, F.: *Transaction Level Modeling with SystemC*. Springer, 2005

[GHW04] Grimm, Ch. ; Heupke, W. ; Waldschmidt, K. *Refinement of Mixed-Signal Systems with Affine Arithmetic.* 2004

[GLMS02] Grötker, T. ; Liaom, S. ; Martin, G. ; Swan, S.: *System Design with SystemC.* Kluwer Academic Publishers, 2002

[GTCD09] Gorissen, D. ; Tommasi, L. D. ; Crombecq, K. ; Dhaene, T.: Sequential Modeling of a Low Noise Amplifier with Neural Networks and Active Learning. In: *Neural Computing and Applications* 18 (2009), Nr. 5, p. 485–494

[Guo07] Guo, Shin-Ming: A Fast Multi-Objective Evolutionary Algorithm for Expensive Simulation Optimization Problems. In: *ICICIC '07: Proceedings of the Second International Conference on Innovative Computing, Informatio and Control.* Washington, DC, USA : IEEE Computer Society, 2007. – ISBN 0–7695–2882–1, p. 324

[Kim06] Kim, Sujin: Gradient-based simulation optimization. In: *WSC '06: Proceedings of the 38th conference on Winter simulation*, Winter Simulation Conference, 2006. – ISBN 1–4244–0501–7, p. 159–167

[Kle08] Kleijnen, J.P.C.: Design Of Experiments: Overview. In: *Winter Simulation Conference*, 2008, p. 479–488

[KO07] Kabirian, Alireza ; Olafsson, Sigurdur: Allocation of simulation runs for simulation optimization. In: *WSC '07: Proceedings of the 39th conference on Winter simulation.* Piscataway, NJ, USA : IEEE Press, 2007. – ISBN 1–4244–1306–0, p. 363–371

[KR03] Key, A. C. ; Rees, L. P.: A sequential-design metamodeling strategy for simulation optimization. In: *Computers & Operations Research 31* Bd. 31, Elsevier, 2003, p. 19111932

[Law07] Law, A.: Statistical analysis of simulation output data: The practical state of the art. In: *Winter Simulation Conference*, 2007, p. 77–83

[MHE08] Markwirth, T. ; Haase, J. ; Einwich, K.: Statistical Modeling with SystemC-AMS for automotive systems. In: *Forum on Specification and Design Languages*, 2008, p. 247–248

[MM06] Mullur, A. A. ; Messac, A.: Metamodeling using extended radial basis functions: a comparative approach. In: *Engineering with Computers* 21 (2006)

[Mon05] Montgomery, D.: *Design and analysis of experiments.* John Wiley & Sons, 2005

[MPLM07] Montevechi, J. A. B. ; de Pinho, A. F. ; Leal, F. ; Marins, F.A.S.: Application of design of experiments on the simulation of a process in an automotive industry. In: *Proceedings of the 2007 Winter Simulation Conference*, 2007

[NYLS05] Nookala, V. ; Ying, C. ; Lilja, D.J. ; Sapatnekar, S.: Microarchitecture-Aware Floorplanning Using a Statistical DoE approach. In: *Design Automation Conference*, 2005, p. 579–584

[NZH+08] Nirmaier, T. ; Zaguirre, J.T. ; Hong, E. ; Spirkl, W. ; Rettenberger, A. ; Schmitt-Landsiedel, D.: Efficient High-Speed Interface Verification and Fault Analysis. In: *IEEE International Test Conference*, 2008

[PAT02] Peterson, Gregory ; Ashenden, Peter ; Teegarden, Darrell: *The System Designer's Guide to VHDL-AMS*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2002. – ISBN 1558607498

[PGZ08] Pelz, G. ; Gergintschew, Z. ; Zeller, C.: Design Quality in the Development of Automotive Smart Power ICs. In: *2. GMM/GI/ITG-Fachtagung*, 2008

[PR10] Pelz, G. ; Rafaila, M.: From Requirements to Comprehensive Verification of Smart Power ICs. In: *Proceedings of the Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, 2010

[RDG+10] Rafaila, M. ; Decker, C. ; Grimm, C. ; Kirscher, J. ; Pelz, G.: Design of Experiments for Reliable Operation of Electronics in Automotive Applications. In: *Forum on specification and design languages*, 2010

[RDGP09a] Rafaila, M. ; Decker, C. ; Grimm, C. ; Pelz, G.: Case Study of High-level Verification of an Automotive Window Lifter ECU. In: *Workshop Multi-Nature Systems: Entwicklung von Systemen mit elektronischen und nichtelektronischen Komponenten*, 2009

[RDGP09b] Rafaila, M. ; Decker, C. ; Grimm, C. ; Pelz, G.: Design of Experiments for Effective Pre-silicon Verification of Automotive Electronics. In: *Forum on specification and design languages*, 2009

[RDGP09c] Rafaila, M. ; Decker, C. ; Grimm, C. ; Pelz, G.: New Methods for System-level Verification using SystemC-AMS Extensions: Application to an Automotive ECU. In: *12. Workshop Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, 2009

[RDGP10a] Rafaila, M. ; Decker, C. ; Grimm, C. ; Pelz, G.: Design of Experiments for Effective Pre-silicon Verification of Automotive Electronics. In: Springer (Hrsg.): *Advances in Design Methods from Modeling Languages for Embedded Systems and SoC's - Selected Contributions from FDL'09*. Springer, 2010 (ISBN-10: 9048193036, ISBN-13: 978-9048193035)

[RDGP10b] Rafaila, M. ; Decker, C. ; Grimm, C. ; Pelz, G.: Sequential Design of Experiments for Effective Model-based Validation of Electronic Control Units. In: *Mikroelektroniktagung ME10*, 2010

[RDGP10c] Rafaila, M. ; Decker, C. ; Grimm, C. ; Pelz, G.: Simulation-based Sensitivity and Worst-Case Analyses of Automotive Electronics. In: *IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2010

[RGDP10] Rafaila, M. ; Grimm, Ch. ; Decker, Ch. ; Pelz, G.: Sequential design of experiments for effective model-based validation of electronic control units. In: *Elektrotechnik und Informationstechnik* 127 (2010), p. 164–170. – 10.1007/s00502-010-0738-x. – ISSN 0932–383X

[San07] Sanchez, S.: Work Smarter, not Harder: Guidelines for Designing Simulation Experiments. In: *Winter Simulation Conference*, 2007

[SB04a] Srinivasaiah, H.C. ; Bhat, N.: Response Surface Modeling of 100nm CMOS Process Technology using Design of Experiments. In: *17th International Conference on VLSI Design*, 2004

[SB04b] Stone, R. ; Ball, J.: *Automotive Engineering Fundamentals*. SAE International, 2004

[SPKA01] Simpson, T. W. ; Peplinski, J. D. ; Koch, P. N. ; Allen, J. K.: Metamodels for Computer-based Engineering Design: Survey and recommendations. In: *Engineering with Computers* 17 (2001), p. 129–150

[SR07] Singhee, A. ; Rutenbar, R.A.: Statistical blockade: a novel method for very fast Monte Carlo simulation of rare circuit events and its application. In: *Design, Automation and Test in Europe*, 2007, p. 1–6

[SREP08] Samii, S. ; Rafiliu, S. ; Eles, P. ; Peng, Z.: A simulation methodology for worst-case response time estimation of distributed real-time systems. In: *Design, Automation and Test in Europe*, 2008, p. 556–561

[SVL07] Sheldon, D. ; Vahid, F. ; Lonardi, S.: Soft-core Processor Customization using the Design of Experiments Paradigm. In: *Design, Automation and Test in Europe*, 2007, p. 1–6

[TM01] Trocine, L. ; Malone, L.C.: An overview of newer, advanced screening methods for the initial phase in an experimental design. In: *Winter Simulation Conference*, 2001, p. 169–178

[TWLBX09] Tate, J. ; Woolford-Lim, B. ; Bate, I. ; Xin, Y.: Comparing Design of Experiments and Evolutionary Approaches to Multi-objective Optimisation of Sensornet Protocols. In: *Congress on Evolutionary Computation*, 2009, p. 1137–1144

[VGE04] Vachoux, A. ; Grimm, C. ; Einwich, K.: Towards Analog and Mixed-Signal SOC Design with SystemC-AMS. In: *IEEE International Workshop on Electronic Design, Test and Applications*, 2004

[VGE05] Vachoux, A. ; Grimm, C. ; Einwich, K.: Extending SystemC to support mixed discrete-continuous system modeling and simulation. In: *IEEE International Symposium on Circuits and Systems*, 2005

[VPB+08] Vasilevski, M. ; Pecheux, F. ; Beilleau, N. ; Abou-shady, H. ; Einwich, K.: Modeling and Refining Heterogeneous Systems With SystemC-AMS: Application to WSN. In: *Design, Automation and Test in Europe*, 2008

[WS07] Wang, G. G. ; Shan, S.: Review of Metamodeling Techniques in Support of Engineering Design Optimization. In: *Journal of Mechanical Design* 129 (2007), Nr. 4, p. 370–380

Internet References

[50] MathWorks. *MATLAB online documentation*. http://www.mathworks.com/help/techdoc/, 2010.

[51] MathWorks. *MATLAB Optimization toolbox online documentation*. http://www.mathworks.com/access/helpdesk/help/toolbox/optim/, 2010.

[52] MathWorks. *MATLAB Statistics toolbox online documentation, regression statistics documentation*. http://www.mathworks.com/help/toolbox/stats/, 2010.

[53] Sigmazone. *Software tool for experiment design*, 2010. http://www.sigmazone.com/doepro--_features.htm.

[54] SystemC-AMS Working Group. *Requirements specification for SystemC Analog Mixed Signal (AMS) Extensions*, 2008. http://www.systemc.org/downloads/standards/.

[55] SystemC Working Group. *SystemC Verification Standard Specification Version 1.0e*, 2003. http://www.systemc.org/downloads/standards/.

[56] Technical University of Denmark. *S. N. Lophaven and H. B. Nielsen and J. Sondergaard: DACE, a MATLAB kriging toolbox*, 2002. http://www2.imm.dtu.dk/~hbn/dace/.

[57] Tilburg University. *Web generator for space filling designs*, 2010. http://www.spacefillingdesigns.nl/.

[58] Wikipedia. *Correlation*, 2010. http://en.wikipedia.org/wiki/Correlation.

[59] Wikipedia. *Importance Sampling*, 2010. http://en.wikipedia.org/wiki/Importance_sampling.

[60] Wikipedia. *Kernel density estimation*, 2010. http://en.wikipedia.org/wiki/Kernel_density_estimation.

[61] Wikipedia. *Symbolic simulation*, 2010. http://en.wikipedia.org/wiki/Symbolic_simulation.

[62] Workshop on C/C++ Modeling for Mixed-Signals Embedded Systems. *G. Noessing: SystemC-AMS Modelling for Voice over IP Physical Interfaces*, 2007. http://www.eas.iis.fraunhofer.de/events/workshops/2007/cmemss/presentations.pdf.

[63] Workshop on C/C++ Modeling for Mixed-Signals Embedded Systems. *W. Scherr: SystemC-AMS Modeling of Embedded Sensors for Automotive Applications*, 2007. `http://www.eas.iis.fraunhofer.de/events/workshops/2007/cmemss/presentations.pdf`.

Glossary

A

ANOVA Analysis of variance: a formal method to identify significant factors, which determines which factor effects are nonzero by analyzing the inequality of response values at different factor treatments, p. 47.

C

CCD Central Composite Design: an experimental design which augments a 2-level factorial design with center points and 2 star points for each factor, p. 41.

CDF Cumulative distribution function: describes the probability that a real-valued random variable X with a given probability distribution will be found at a value less than its argument x. It is computed by integrating the probability density function: $F_X(x) = \int_{-\infty}^{x} PDF_X(t)dt$, p. 36.

D

DoE Design of Experiments: discipline aimed at improving the outputs of experiments, by applying mathematical statistics on the experiment planning and analysis. Experimental design: output of the design of the experiment, specifies in a matrix format the settings for the experiment inputs, p. 7.

DUT Device under test: system under verification. Since the verification is simulation-based, the system is represented by its model. For each simulation, it must be set up in a test bench, configured and stimulated according to the test case, p. 22.

E

ECU Electronic control unit: an embedded system that controls one or more of the electrical systems or subsystems in a motor vehicle, p. 1.

G

GA Genetic algorithm: a heuristic that addresses optimization and search problems, using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover, p. 18.

L

LHS Latin Hypercube Sampling: a space filling experimental design which generates samples in order to ensure that each region of the factor space is equally represented, p. 15.

M

Monte-Carlo Monte-Carlo (or random test methods): a method of circuit- or system-level analysis, which randomizes prior to the simulation design parameters to determine their effects on the simulation outputs, p. 11.

P

PDF Probability density function: a function that describes the relative likelihood for a continuous random variable to occur at a given point in the observation space. The probability of a random variable falling within a given range is given by the integral of its density over the set, p. 36.

R

R3FF Resolution 3 fractional factorial: a fraction of the full $2^{number\_of\_factors}$ factorial design which estimates main factor effects aliased on 2-factor interactions. It is commonly used for screening, because it can extract preliminary main effects, assumed significant compared to the interactions, p. 39.

R5FF Resolution 5 fractional factorial: a fraction of the full $2^{number\_of\_factors}$ factorial design which estimates main factor effects and 2-factor interactions, but 2-factor interactions are aliased with 3-factor interactions, p. 39.

RFDoE Random-factor experimental design: a type of experimental design which extends random test methods by handling factor correlations and adding postprocessing steps to address sensitivity and worst-case analysis, p. 17.

S

SoC System-on-a-chip: an electronic system with all components contained into a single integrated circuit (chip), p. 1.

Systems-in-a-Package System-in-a-Package: a number of integrated circuits enclosed in a single package or module, p. 1.

T

TLM Transaction-level modeling : a high-level approach to modeling digital systems where details of communication among modules are separated from the details of the implementation of functional units or of the communication architecture, p. 4.

Abbreviations

ADC	Analog to digital converter
ANOVA	Analysis of variance
CCD	Central Composite Design
CDF	Cumulative distribution function
Cfg&Stim	Configuration and Stimuli unit
Ctrl	Controller
DAC	Digital to analog converter
DMOS	Double diffused metal oxide semiconductor
DoE	Design of Experiments
DUT	Device under test
ECU	Electronic control unit
GA	Genetic algorithm
LHS	Latin Hypercube Sampling
LIN	Local Interconnect Network
LSS	Low Side Switch
HSS	High Side Switch
MCU	Microcontroller subsystem
MI	Measurement Interface
MoCC	Models of computation and communication
NaN	Not-a-number
OPAMP	Operational Amplifier
PDF	Probability density function
PID	Proportional integral derivative controller
PWL	Piece-wise linear
R3FF	Resolution 3 fractional factorial
R4FF	Resolution 4 fractional factorial
R5FF	Resolution 5 fractional factorial
RFDoE	Random-factor experimental design
SoC	System-on-a-chip
SPI	Serial Peripheral Interface
TLM	Transaction-level modeling

List of Figures

List of Tables

Monica Rafaila

Address: Tumblingerstr. 56
80337 Munich, Germany

Date of birth: 08/08/1983
Email: monicarafaila@yahoo.com
Telephone: +49 176 25774431

EDUCATION

PhD. 10.2007 - expected 10.2010

Technical University of Vienna
Institute for Computer Technology, coordinator: Univ.Prof. Dr.phil.nat. Christoph Grimm
The work focuses on the planning and analysis of simulation experiments, for automotive electronic control units. The number of simulation runs necessary to deal with a high number of sources of variation is reduced, while a good verification coverage is maintained. Design of Experiments and Metamodelling methods are intensively used.

Dipl.Ing. 10.2002 - 07.2007

Polytechnic University of Bucharest
Faculty of Electronics, Telecommunications and Information Technology
Networks and Software for Telecommunications Division
Graduation: grade 10.00 (the Romanian scale is increasing from 1 to 10), top in the year.
The topic of the Bachelor's Diploma was Virtual Private Networks (Layer 2 vs. Layer 3).

Certificates

- Fundamentals of Java Programming (Cisco Networking Academy certified)
- Fundamentals of Unix (Cisco Networking Academy certified)
- Cisco Certified Network Associate (Cisco Networking Academy certified) I (Networking Basics), II (Routers and Routing Basics)
- Symbolic Model Checking (Technical University of Vienna)

Reviewed conference publications

- M. Rafaila, C. Decker, C. Grimm, G. Pelz: "Simulation-based Sensitivity and Worst-Case Analyses of Automotive Electronics"; *IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems, Vienna, 2010.*
- M. Rafaila, C. Decker, C. Grimm, G. Pelz: "Sequential Design of Experiments for Effective Model-based Validation of Electronic Control Units"; *Mikroelektronik Tagung, Vienna, 2010.*
- M. Rafaila, C. Decker, C. Grimm, G. Pelz: "Design of Experiments for Effective Presilicon Verification of Automotive Electronics"; *Forum of Design Languages, Nice, 2009.*
- M. Rafaila, C. Decker, C. Grimm, G. Pelz: "New Methods for System-level Verification using SystemC-AMS Extensions: Application to an Automotive ECU"; *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen, Berlin, 2009.*
- M. Rafaila, C. Decker, C. Grimm, G. Pelz: "Case Study of High-level Verification of an Automotive Window Lifter ECU"; *Workshop Multi-Nature Systems: Entwicklung von Systemen mit elektronischen und nichtelektronischen Komponenten, Ulm, 2009.*

Articles and book chapters

- M. Rafaila, C. Decker, C. Grimm, G. Pelz: "Sequential Design of Experiments for Effective Model-based Validation of Electronic Control Units"; *published in the journal: Elektrotechnik und Informationstechnik, Springer Vienna, volume 127, issue 6, pages 164-170,* `http://dx.doi.org/10.1007/s00502-010-0738-x`, 2010.
- M. Rafaila, C. Decker, C. Grimm, G. Pelz: "Design of Experiments for Effective Presilicon Verification of Automotive Electronics"; *published in the book: "Advances in Design Methods from Modeling Languages for Embedded Systems and SoC's", Springer Netherlands, series "Lecture Notes in Electrical Engineering", chapter 9, pages 141-158,* `http://dx.doi.org/10.1007/978-90-481-9304-2_9`, 2010.

Invited papers and presentations

- M. Rafaila, C. Decker, C. Grimm, J. Kirscher, G. Pelz: "Design of Experiments for Reliable Operation of Electronics in Automotive Applications"; *Forum of Design Languages, Southampton, 2010 (invited paper*).
- G. Pelz, M. Rafaila: "From Requirements to Comprehensive Verification of Smart Power ICs"; *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen, Dresden, 2010 (invited paper).*
- M. Rafaila, C. Grimm, C. Decker, G. Pelz: "SystemC-AMS High-level Verification of Automotive Applications, in the Presence of Uncertainties"; *Forum of Design Languages, Stuttgart, 2008 (invited presentation).*

Prizes

- Best paper award, Mikroelektronics Tagung (Embedded Systems track), 2010, Vienna.
- Prizes in the Signals, Circuits and Systems Section of Tudor Tanasescu Annual National Contest, 2^{nd} (2005) and 4^{th} places (2006), Polytechnic University of Bucharest.
- 2^{nd} and 3^{rd} prizes in the Romanian National Olympics of Mathematics (1998 - 2001).

EMPLOYMENT

Infineon Technologies, Munich 10.2007 - present

Automotive Power Department, Design Methodologies Division

Responsibilities

- Concept-level modelling of electronic control units.
- Tasks in the AutoSUN research project:
  - Develop and test methods for abstract modelling and efficient simulation-based verification.
  - Provide an interface to collaborators.
  - Document and present results to the funding authorities.

2kTelecom, Bucharest 07.2006 - 07.2007

Technical Department, Network Operating Center

Responsibilities

- Provide level 1 and 2 technical support and troubleshooting.
- Monitor and optimize network parameters.

SKILLS

Technical

- Hardware Description languages: SystemC, SystemC-AMS, SystemC TLM1.0, VHDL.
- Programming languages: MATLAB, C++, Java.
- IDEs: Eclipse, Keil (C, assembler for the 8051 core), Netbeans (Java).

Operating systems: UNIX/Linux.