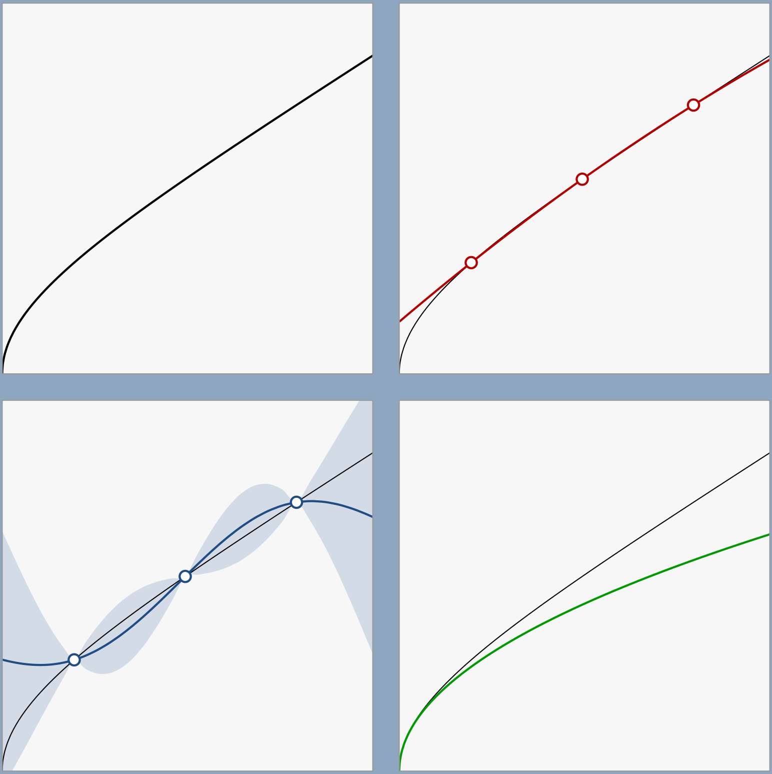

Michael Sinsbeck

Uncertainty Quantification for Expensive Simulations - Optimal Surrogate Modeling under Time Constraints

Uncertainty Quantification for Expensive Simulations

Optimal Surrogate Modeling under Time Constraints

Von der Fakultät Bau- und Umweltingenieurwissenschaften der Universität Stuttgart zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von

Michael Sinsbeck

aus Mülheim an der Ruhr

Hauptberichter:	Prof. Dr.-Ing. Wolfgang Nowak
Mitberichter:	Prof. Dr.-Ing. Rainer Helmig
	Prof. Daniel Tartakovsky

Tag der mündlichen Prüfung: 6. September 2016

Institut für Wasser- und Umweltsystemmodellierung
der Universität Stuttgart
2017

Für Großmutter und Großvater

Contents

Symbols and Abbreviations

Latin letters

d	dimension of the parameter domain Ω
F_X	cumulative distribution function of variable X
K	hydraulic conductivity
L	likelihood
n	number of nodes
N	number of Monte-Carlo realizations,
	also number of discrete nodes of a distribution
p	number of ansatz functions $p = \dim \mathbb{P}$,
	also number of expansion terms
t	number of test functions $t = \dim \mathbb{T}$
u	model function
U	random field describing u
$\boldsymbol{x}$	vector of nodes $\boldsymbol{x} = [x_1, \dots, x_n] \in \Omega^n$
X	input parameter
Y	model output $Y = u(X)$
Z	observable data

Greek letters

ε	measurement error
θ	water content
$\Lambda(\mathcal{P})$	Lebesgue constant of operator $\mathcal{P}$
μ	measure of X
ξ, ζ	location (spatial coordinates)
π_ε	probability density function (pdf) of random variable ε
π	(without index): prior pdf of X
$\hat{\pi}$	(without index): posterior pdf of X
τ	time (temporal coordinate)
ψ	pressure head
Ω	parameter domain, $\Omega \subseteq \mathbb{R}^d$

Stochastic operators

$P[A]$	probability of event A
$\mathrm{E}[\cdot]$	expected value of a random variable
$\mathrm{Var}[\cdot]$	variance of a random variable
$\mathrm{Cov}[\cdot,\cdot]$	covariance of two random variables

Spaces

$\mathbb{P}$	ansatz space
$\mathbb{T}$	test space
$\mathbb{U}$	space containing u

Distributions

$\mathcal{N}\left(m,\sigma^2\right)$	normal distribution with mean m and variance σ^2
$\mathcal{U}(A)$	uniform distribution on the set A

Modifiers

$\hat{\pi}$	hat	posterior of π
$x^\star$	star	optimal x
$\tilde{x}$	tilde	approximation of x

Abbreviations

BU	Bayesian updating
cdf	cumulative distribution function
DoE	design of (computer) experiments
gpe	Gaussian process emulator
i.i.d.	independent and identically distributed (random variables)
MC	Monte-Carlo
MCMC	Markov-Chain Monte-Carlo
MLMC	Multi-Level Monte-Carlo
OSC	optimized stochastic collocation
PCE	polynomial chaos expansion
pdf	probability density function
QoI	quantity of interest
SC	stochastic collocation
UP	uncertainty propagation
UQ	uncertainty quantification

Abstract

Motivation and Goal

Computer simulations allow us to predict the behavior of real-world systems. Any simulation, however, contains imperfectly adjusted parameters and simplifying assumptions about the processes considered. Therefore, simulation-based predictions can never be expected to be completely accurate and the exact behavior of the system under consideration remains uncertain. The goal of uncertainty quantification (UQ) is to quantify how large the deviation between the real-world behavior of a system and its predicted behavior can possibly be. Such information is valuable for decision making.

Computer simulations are often computationally expensive. Each simulation run may take several hours or even days. Therefore, many UQ methods rely on surrogate models. A surrogate model is a function that behaves similarly to the simulation in terms of its input-output relation, but is much faster to evaluate. Most surrogate modeling methods are convergent: with increasing computational effort, the surrogate model converges to the original simulation. In engineering practice, however, results are often to be obtained under time constraints. In these situations, it is not an option to increase the computational effort arbitrarily and so the convergence property loses some of its appeal. For this reason, the key question of this thesis is the following:

What is the best possible way of solving UQ problems
if the time available is limited?

This is a question of optimality rather than convergence. The main idea of this thesis is to construct UQ methods by means of mathematical optimization so that we can make the optimal use of the time available.

Contributions

This thesis contains four contributions to the goal of UQ under time constraints.

1. A widely used surrogate modeling method in UQ is stochastic collocation, which is based on polynomial chaos expansions and therefore leads to polynomial surrogate models. In the first contribution, I developed an optimal sampling rule specifically designed for the construction of polynomial surrogate models. This sampling rule showed to be more efficient than existing sampling rules because it is stable, flexible and versatile. Existing methods lack at least one of these properties. Stability guarantees that the response surface will not oscillate between the sample points, flexibility allows the modeler to choose the number of function evaluations freely, and versatility means that the method can handle multivariate input distributions with statistical dependence.

2. In the second contribution, I generalized the previous approach and optimized both the sampling rule and the functional form of a surrogate in order to obtain a general optimal surrogate modeling method. I compared three possible approaches to such optimization and the only one that leads to a practical surrogate modeling method requires the modeler to describe the model function by a random field. The optimal surrogate then coincides with the Kriging estimator.

3. I developed a sequential sampling strategy for solving Bayesian inverse problems. Like in the second contribution, the modeler has to describe the model function by a random field. The sequential design strategy selects sample points one at a time in order to minimize the residual error in the solution of the inverse problem. Numerical experiments showed that the sequential design is more efficient than non-sequential methods.

4. Finally, I investigated what impact available measurement data have on the model selection between a reference model and a low-fidelity model. It turned out that, under time constraints, data can favor the use of a low-fidelity model. This is in contrast to model selection without time constraint where the availability of data often favors the use of more complex models.

Conclusions

From the four contributions, the following overarching conclusions can be drawn.

- Under time constraints, the number of possible model evaluations is restricted and the model behavior at unobserved input parameters remains uncertain. This type of uncertainty should be taken into account explicitly. For this reason, random fields as surrogates should be preferred over deterministic response surface functions when working under time constraints.
- Optimization is a viable approach to surrogate modeling. Optimal methods are automatically flexible which means that they are easily adaptable to the computing time available.
- Under time constraints, all available information about the model function should be used.
- Model selection with and without time constraints is entirely different.

Kurzfassung

Motivation und Ziel

Mit Hilfe von Computersimulationen können wir das zukünftige Verhalten von Systemen vorhersagen. In jeder Simulation stecken allerdings vereinfachende Modellannahmen und die benötigten Eingabeparameter sind oft nicht genau bekannt. Daher sind simulationsbasierte Vorhersagen niemals exakt und das tatsächliche zukünftige Verhalten des betrachteten Systems bleibt unsicher. Das Ziel von Unsicherheitsquantifizierung (UQ) ist es die mögliche Abweichung zwischen Vorhersage und tatsächlichem Verhalten zu quantifizieren.

Computersimulationen benötigen oft lange Rechenzeiten von mehreren Stunden oder sogar Tagen. Daher verwenden viele UQ-Methoden Surrogatmodelle. Ein Surrogatmodell ist eine mathematische Funktion, die sich bezüglich der Input-Ouput-Beziehung so ähnlich verhält wie die eigentliche Simulation, dabei aber viel schneller auswertbar ist. Die meisten Methoden zur Konstruktion von Surrogatmodellen sind konvergent. Das heißt, dass das Surrogatmodell mit steigendem Rechenaufwand gegen die Simulation konvergiert. In der Praxis steht allerdings oft nur eine begrenzte Rechenzeit zur Verfügung und daher kann nicht jede beliebig hohe Genauigkeit erreicht werden. Die zentrale Frage dieser Arbeit ist daher:

Wie sollten UQ-Probleme am besten gelöst werden,
wenn die Rechenzeit beschränkt ist?

Mit dieser Frage suchen wir also nach *optimalen* UQ-Methoden. Das Ziel dieser Arbeit ist es UQ-Methoden mit Hilfe von mathematischer Optimierung zu entwickeln, so dass die gegebene Zeit optimal ausgenutzt wird.

Beiträge

Diese Arbeit liefert vier Beiträge zu dem genannten Ziel.

1. Eine der am weitesten verbreiteten Surrogatmodellmethoden ist die stochastische Kollokation, beruhend auf der polynomiellen Chaosentwicklung. Im ersten Beitrag greife ich diese bestehende Methode auf und optimiere einen Teil davon: die Samplingregel. Die entwickelte, optimale Samplingregel ist speziell dafür geeignet, polynomielle Surrogatmodelle zu konstruieren. In numerischen Experimenten erreicht sie eine bessere Effizienz als bestehende Samplingregeln.
2. Im zweiten Beitrag verallgemeinere ich den Ansatz aus dem ersten Beitrag, so dass neben der Samplingregel auch noch die Funktionsform des Surrogatmodells optimiert wird. Da hier mehr Spielraum in der Wahl der Zielwertfunktion besteht, vergleiche ich drei verschiedene Formulierungen einer solchen Optimierung. Die einzige Formulierung, die zu einer praktikablen Surrogatmodellmethode führt, setzt voraus, dass die Modellfunktion mit einem Zufallsfeld beschrieben wird. Das optimale Surrogat, das sich dann ergibt, stimmt mit dem Kriging-Schätzer überein, der aus der Geostatistik bekannt ist.
3. Im dritten Beitrag stelle ich eine sequentielle Samplingmethode für die Lösung von Bayes'schen inversen Problemen vor. Wie im zweiten Beitrag muss die Modellfunktion dazu mit einem Zufallsfeld beschrieben werden. Das sequentielle Design wählt dann im Parameterraum die Auswertungspunkte nacheinander so aus, dass jeweils der erwartete Fehler in der Lösung des inversen Problems minimiert wird. Das sequentielle Design erreicht in numerischen Experimenten bei gleicher Rechenzeit genauere Ergebnisse als bestehende, nicht-sequentielle Samplingmethoden.
4. Im vierten Beitrag gehe ich der Frage nach, welchen Einfluss verfügbare Messdaten auf die Modellwahl zwischen einem Referenzmodell und einem vereinfachten Modell haben. Anhand eines Infiltrationsproblems zeige ich, dass unter Zeitbeschränkung die Verfügbarkeit von Messdaten die Nutzung eines vereinfachten Modells begünstigen kann. Dieses Ergebnis steht in einem interessanten Gegensatz zu dem üblichen Modellwahlproblem ohne Zeitbeschränkung: Dort ist die Tendenz oft anders herum und die Verfügbarkeit von mehr Messdaten spricht eher für die Nutzung eines komplexeren Modells.

Schlussfolgerungen

Aus den präsentierten Beiträgen können die folgenden Schlussfolgerungen gezogen werden:

- Unter Zeitbeschränkungen ist die Anzahl der Modellauswertungen beschränkt und daher ist das Verhalten des Modells für unbeobachtete Eingabeparameter unsicher. Diese Art von Unsicherheit sollte explizit berücksichtigt werden, z.B. indem Zufallsfelder statt deterministischer Surrogatmodelle verwendet werden.
- Mathematische Optimierung ist als Methode zur Konstruktion von Surrogatmodellmethoden geeignet. Optimale Methoden sind automatisch auch flexibel, d.h. sie lassen sich leicht an die gegebene Rechenzeit anpassen.
- Unter Zeitbeschränkungen ist es besonders wichtig alle verfügbaren Informationen über die Modellfunktion zu verwenden.
- Unter Zeitbeschränkungen verhält sich das Modellwahlproblem komplett anders als ohne Zeitbeschränkungen.

Part I.

Prologue

1. Introduction

Computer simulations allow us to reproduce and predict the behavior of real-world systems. Any simulation, however, contains simplifying assumptions about the processes considered and, moreover, input parameters are often not known precisely. Therefore, we can never expect simulation-based predictions to be completely accurate. George E. P. Box captured this fact in a statement that has become iconic among modelers [16, p. 424]:

Essentially, all models are wrong, but some are useful.

Every simulation is based on a model so, as a direct consequence, all simulation results are also wrong. With that said, the good news is that simulations can still be useful. To decide whether a specific simulation-based prediction is useful, we need to have an idea about *how wrong* it is. This is what the field of *uncertainty quantification* (UQ) is about. The deviation between the real-world behavior of a system and its predicted behavior is uncertain, and ideally, we would like to quantify how large this deviation can possibly be.

There is a number of sources for uncertainty and, throughout this thesis, I will only consider one of them: parameter uncertainty. With this premise, the two most important UQ tasks are uncertainty propagation and Bayesian updating. In Section 1.1, I will give a short overview of the possible sources of uncertainty, explain why parameter uncertainty is important and provide more details on these two tasks.

A very simple approach to UQ problems is to conduct a Monte-Carlo simulation: instead of running a simulation only once, we run it multiple times with different input parameters. As a result, we obtain a collection of predictions and from these we can make a statement about the uncertainty in the prediction. In practice, however, there is a catch: simulations are often *computationally expensive*. That means that each simulation run takes several hours or even days and so, it is practically impossible to run a proper Monte-Carlo simulation with multiple thousands of individual simulation runs. More consequences of using expensive simulations are briefly presented in Section 1.2. Unless noted otherwise, I will use the term “expensive” not for monetary costs, but for computational costs which refer to the computing time required.

When dealing with computationally expensive simulations in UQ, a common strategy is to use *surrogate models*. A surrogate model is a function that behaves similarly to the simulation in terms of its input-output relation, but that is much faster to evaluate. I will give an overview of common surrogate modeling methods in Section 1.3.

Most surrogate modeling methods focus on the property of convergence. This means that, with increasing computational effort, the surrogate model converges to the original simulation, and so the uncertainty can be quantified with any desired accuracy. In engineering practice, however, results are often to be obtained under *time constraints*. In these situations, it is not an option to further increase the computational effort and therefore the convergence property loses some of its appeal. For this reason, the key question of this thesis is the following:

What is the best possible way of solving UQ problems
if the time available is limited?

This is a question of *optimality* rather than convergence. The aspect of optimality is central to this thesis and all of my contributions aim at optimality in one way or another. To my knowledge, time constraints and optimality have not yet been explicitly considered in UQ.

In summary, my work is motivated by the need to quantify uncertainties in simulation-based predictions. The goal is to develop surrogate modeling methods that account for a possible time constraint. The approach is to construct methods by means of optimization. In the following three sections (1.1 to 1.3), I will provide more background information on the aspects of uncertainty quantification, expensive simulations and surrogate modeling. Then, in Section 1.4, I will formulate three hypotheses about the consequences of time constraints and, in Section 1.5, I will give an overview of the structure of the thesis.

1.1. Uncertainty Quantification

To define the scope of this thesis more precisely, it is useful to classify the uncertainty under consideration in terms of its *location* and *level*. Both terms go back to work by Walker et al. [153]. Afterwards, I provide a brief description of the two UQ tasks named earlier: uncertainty propagation and Bayesian updating.

Location of Uncertainty

The process of setting up a simulation involves multiple steps. The term "location of uncertainty" refers to which of these steps causes uncertainty (this is what I called source of uncertainty in the introductory section). A typical simulation procedure consists of the following steps:

1. The modeler chooses what system to simulate and what aspects to include in the simulation. These choices constitute the context.
2. The system at hand is described by mathematical equations or relationships. This defines the model structure.
3. All numerical parameter values in the model are specified. Some authors differentiate between model parameters (which are properties of the system at hand, such as physical constants) and model input (which are specific to one scenario or test case, such as external driving forces or material parameters) [153]. Throughout this thesis, I will not make this differentiation and will call all of these numerical values *model inputs*.
4. Finally, the mathematical equations are solved on a computer. In this step, the original equations are typically discretized and then solved using numerical schemes.

This list of steps is loosely based on the work of Walker et al. [153]. Each of these steps is one potential source of uncertainty. If an error is made in one of the steps, then the final simulation result will also be wrong. In this thesis, I will only consider parameter uncertainty, i.e. uncertainties in model inputs, see step 3 in the procedure above. The final vision of UQ as a scientific field should, of course, be to consider all possible uncertainty sources simultaneously. My contributions have to be understood as one piece of the puzzle in this larger context. Once each individual source of uncertainty is well understood and can be handled efficiently, the next step has to be to consider them together.

To keep the terminology clear, I will summarize all numerical values required in a simulation with the term "(model) inputs", and call those inputs that are uncertain "parameters".

Assuming that the other sources of uncertainty are negligible directly implies the following three assumptions:

- all relevant aspects are included in the model;
- we know mathematical equations that describe the real-world system exactly;
- we can solve these equations without errors.

These assumptions mean that, for a given input, the simulation at hand returns the physically correct output and that the simulation itself is not a source of additional uncertainty. We capture the input-output relationship of the simulation in a mathematical function, called the *model function*. This function is a black-box function in the following sense: for any input, the output can be computed by running the simulation, but we are not able to write down the model function as a closed-form mathematical expression.

I argue that parameter uncertainty is an important source of uncertainty because it is present in almost all simulations. Two examples are:

Material Parameters Material parameters are determined by measurements. These, however, are subject to measurement errors and are often monetarily expensive. It is especially difficult to obtain a complete description of a material's parameters, when the material is not homogeneous so that the material parameters vary over all three spatial dimensions. Moreover, if the material is solid (such as soil in the subsurface), then it is physically not always possible to measure all material parameters without destroying the material itself. As a result, the available knowledge about material parameters is often sparse and not fully accurate.

External Driving Forces External driving forces, such as the precipitation in a rainfall event, are often not under the control of the modeler. Such driving forces are naturally uncertain.

Of course, there are more possible uncertain parameters, such as boundary conditions, initial conditions or geometric specifications.

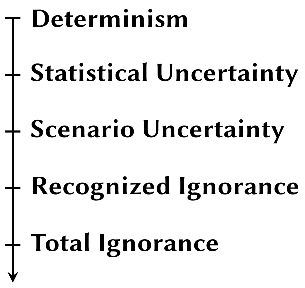

Figure 1.1.: Levels of uncertainty, based on Walker et al. [153]

Level of Uncertainty

The level of uncertainty describes how much we know about a quantity [153]. The different levels range from determinism ("we know the quantity exactly") to total ignorance ("we know nothing about the quantity and we are not even aware of our lack of knowledge"). All intermediate levels are shown in Figure 1.1.

Throughout this thesis, I will assume that all uncertainty is on the level of *statistical uncertainty*. That means that, for any parameter, we can state our uncertainty about its value in terms of a probability distribution which is basically a list of all possible values of the parameter together with probabilities. Mathematically, parameters are then described by random variables. A more detailed discussion of the model parameters will be given in Section 2.1.1. To keep the terminology clear, I will call a quantity *uncertain* if it has a known probability distribution, and *unknown* if it does not have a known probability distribution.

Parameters play such an important role in this thesis that I use the variables x, y, z and X, Y, Z only for parameters, and denote spatial coordinates by ξ and ζ (instead of the usual x and z).

Tasks

Under the premise of statistical parameter uncertainty, the two main tasks in UQ are uncertainty propagation (UP) and Bayesian updating (BU).

Uncertainty Propagation Given uncertainty in the simulation inputs, UP is the investigation of the model outcome uncertainty: how does the

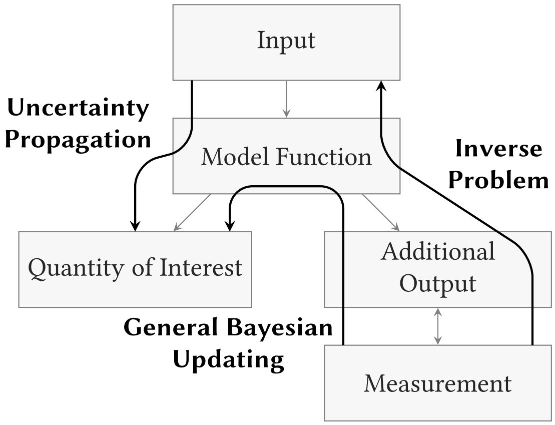

Figure 1.2.: The flow of information in the main UQ tasks of uncertainty propagation and Bayesian updating. The latter comes in the two variants “inverse problem” and “general Bayesian updating”.

uncertainty propagate through the model function? Depending on how sensitive the output is to the specific input parameters, propagated uncertainty might be larger or smaller than the parameter uncertainty. UP is also called forward propagation of uncertainties.

Bayesian Updating BU is the process of using measurement data or observations to update the knowledge about unobserved quantities. I will distinguish between (Bayesian) inverse problems and general Bayesian updating. In inverse problems, a measurement is used to derive information about the simulation inputs. This procedure is also called an inverse uncertainty quantification. With the term general Bayesian updating, I refer to the process of observing one output quantity to draw conclusions about another output quantity.

Figure 1.2 shows the flow of information in these two tasks. Mathematical details of the UQ tasks are presented in Chapter 2.

1.2. Expensive Simulations and Time Constraints

I assume that the model function is computationally expensive, which means that each function evaluation takes a considerable amount of computing time. Moreover, I assume that all other calculations, such as pre- or post-processing, are fast compared with the model function, so that their computational costs can be neglected and the overall computational costs can be measured purely in terms of the number of model evaluations.

The goal of this work is to develop UQ methods for expensive simulations and then to test, whether they work as expected. A convenient consequence of measuring computational costs in terms of model evaluations is that we can do these tests without actually using expensive simulations. If two simulations have similar properties in general and only differ in terms of their computational costs, then the efficiency of a UQ method with respect to the number of model evaluations will be the same for both simulations. So for testing newly developed methods, there is no need to employ an expensive simulation. The use of an expensive simulation would actually be disadvantageous for testing: to assess the performance of a UQ method, we need to compare its solution to a reference solution. Such reference solution can only be computed if the simulation itself is fast. For these reasons, all numerical examples in this thesis are based on relatively fast simulations that have run times in the range of seconds.

Once a new method is tested, it should, of course, be applied to problems with large computing times. With this goal in mind, we have to consider two important aspects: time constraints and code uncertainty.

Time Constraints

In practice, the computing time available is limited, for example in the form of a project deadline. We assume that the modeler knows in advance how many function evaluations she can afford. She is interested in getting the best result possible in the time available.

To provide a rough number, let us assume that the maximum number of function evaluations is about 100 or fewer.

A fixed time constraint calls for *flexible* numerical methods. I call a method flexible if the modeler can easily adapt the required run time to the time available. An example of a flexible method is a Monte-Carlo simulation: the modeler only needs to divide the time available by the time required for one simulation to obtain the number of possible realizations. An example for an inflexible method is the solution of a partial differential equation on a grid discretization that can only be refined globally. Let us assume the solution on a coarse grid takes 1 hour of computing time. If the grid is refined by factor 2 in all three dimensions, then the new grid has 8 times as many nodes as the coarse grid and the solution would take more than 8 hours. If the time available is 7 hours, we could only calculate the solution on the coarse grid, which means that the method would not make good use of the available time.

Code Uncertainty

The assumption of an expensive, black-box model function together with parameter uncertainty leads to a new type of uncertainty: code uncertainty. It denotes the uncertainty about the input-output relationship of the model function at points in the parameter space that have not been evaluated. The term was first mentioned by Kennedy and O'Hagan [75]. It is not to be confused with numerical uncertainty, which refers to uncertainty induced by an inaccurate numerical solution of the model equations (which we neglect according to our assumptions).

Code uncertainty does not exist in deterministic simulations with known parameters: here, the simulation can simply be run for the known parameters and there is no uncertainty about the output. But with parameter uncertainty, this is different. The model function is to be evaluated on a spectrum of possible parameter values, but we can only afford a finite number of function evaluations. At all parameter values between those function evaluations, the actual behavior of the model function remains uncertain. For this reason, we may say that the response of the model function with respect to changes in the input is uncertain. Some of the methods presented in this work explicitly incorporate the notion of code uncertainty.

1.3. Surrogate Modeling

The idea of surrogate modeling is to replace the original simulation by a function that behaves similarly, but is faster. This function is the surrogate model. In my work, I use three surrogate modeling methods as the main tools for UQ: response surface methods, emulators and low-fidelity models. In this section, I give a short overview of these three methods and point out their main differences from a practical point of view. In Chapter 3, I present the mathematical aspects of the three methods.

Response Surface Methods A response surface is a mathematical function fitted to the response of the model function. First, the modeler selects a class of parametric functions (e. g. polynomials of degree 2), and then determines the function parameters (e. g. the polynomial coefficients) using the information from a finite number of evaluations of the model function. The model function is treated as a black box and the response surface is constructed only from the input-output pairs available from the evaluations. In that sense, response surface methods are data-driven and nonintrusive. I only consider linear methods, i. e., methods where the relationship between the model function and the response surface is a linear operator. This does not imply that the response surface has to be a linear function of the model parameters. Existing response surface methods are, for example, polynomial interpolation/regression, radial basis functions [18] and artificial neural networks [13]. In UQ, the most commonly used response surface method is stochastic collocation (SC), which is a polynomial interpolation/regression method based on polynomial chaos expansions (PCE). SC will be presented in detail in Section 3.1.2.

Emulators When a random field is used as a surrogate model, then it is called an *emulator* [109]. A random field is a stochastic description of an uncertain function. It can be thought of as a collection of infinitely many functions together with probabilities, and the model function is assumed to be one of these functions.

Emulators are data-driven as well: after a general shape of the random field has been decided on, it is adapted to a number of input-output pairs from the model function. This is done by keeping only those functions in the emulator that interpolate the model response. All other functions are removed. This step is called conditioning or Kriging. Mathematical details will be provided in Section 3.2.

The collection of functions can be used to form a response surface (by averaging them) or to express code uncertainty (by considering the variance between the individual functions). As such, an emulator can be thought of as a response surface together with stochastic error bars and these error bars are an expression of code uncertainty.

Low-Fidelity Models A low-fidelity model is a reduced version of the model function. Often, a simulation can be made faster by either reducing the spatial or temporal resolution or neglecting some physical aspects in the model. Unlike the previous two surrogate modeling methods, low-fidelity models are not data-driven. The construction of a low-fidelity model is not done by observing the input-output relation of the model function, but by choosing simpler model equations or faster numerical schemes.

Please note that I use the term “surrogate model” as an umbrella term for any function that may replace a simulation. In the literature, this term is sometimes more narrowly defined and only denotes what I call “response surface” here. In my terminology, every response surface is a surrogate model, but surrogate models also include emulators and low-fidelity models.

Furthermore, the term “emulator” is adopted from O’Hagan’s terminology in which an emulator is a random field that is used as a surrogate [109]. Other authors use the term “emulator” synonymously with “response surface” or “surrogate model”.

Schematic Comparison

Figure 1.3 provides a schematic overview of the three surrogate modeling methods.

In the upper left plot, a model function is shown. In this example, the parameter on the x-axis is the height from which an object is dropped. The model function returns the time until the object reaches the ground. The underlying model assumes a drag force proportional to the squared velocity. If the plot is rotated 90 degrees clockwise, it shows the trajectory of a falling object (height as a function of time). For small heights, the model function is a square root function because the drag force is negligible at early times and so the trajectory is almost a parabola. For larger heights, the trajectory becomes more and more linear, as the object reaches its terminal velocity.

In the upper right plot, a polynomial response surface model of degree 2 is shown. It was constructed by interpolating the model response at three

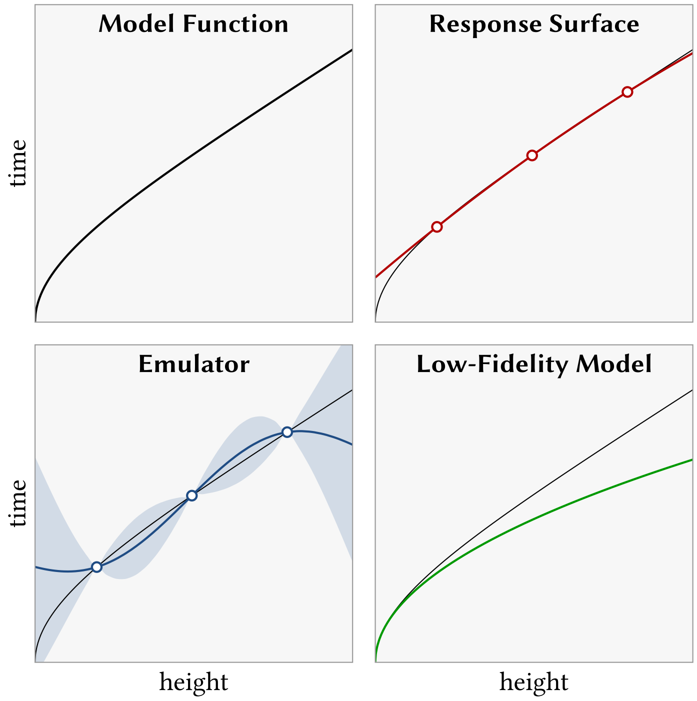

Figure 1.3.: Schematic comparison of the three tools

different heights. Overall, the response surface very closely resembles the model function. The only large deviation lies in the bottom left part of the plot, where the surrogate becomes unphysical and predicts a positive time even if the height is zero. This error shows that a polynomial response surface is merely a mathematical tool which does not incorporate physical knowledge.

In the lower left plot, an emulator is shown. It is conditioned to the same three points as the surrogate in the second plot. The solid line indicates the mean function and the shaded area delineates one standard deviation around the mean. The mean function deviates from the model function in some parts of the plot, but the method is aware of these deviations, as indicated by the shaded area. Similar to response surfaces, emulators do not necessarily include physical knowledge and therefore run the risk of predicting unphysical behavior.

Finally, the lower right plot shows the output from a low-fidelity model. In this example, the drag force is neglected, resulting in a purely parabolic trajectory. For small heights, the low-fidelity model is very accurate. However, with larger heights, the error grows as the effect of the neglected force becomes more and more visible. Low-fidelity models are often constructed by careful modeling decisions and therefore contain at least an approximation of the available physical knowledge. In this example, the low-fidelity model correctly predicts a fall time of zero for a height of zero.

1.4. Hypotheses

As I mentioned earlier, the main novelty of my work is that I consider time constraints in UQ. Regarding the consequences of time constraints, I propose the following three hypotheses.

1. Under time constraints, code uncertainty plays an important role. It should be taken into account explicitly, for example by using emulators.
2. Under time constraints, optimization is a viable approach to surrogate modeling. Optimal methods are flexible automatically.
3. Under time constraints, all available information about the model function should be used.

At the end of this thesis, in Chapter 8, I will come back to these hypotheses and discuss them in the light of the results presented throughout the thesis.

1.5. Structure of the Thesis

This thesis is divided into three parts: Prologue, Contributions and Epilogue.

Prologue Besides this introduction, the prologue contains an overview of the two UQ tasks in Chapter 2 (“Tasks”) and an overview of the different surrogate modeling methods in Chapter 3 (“Tools”). The prologue is mostly a summary of the state of the art and as such contains almost no contributions from myself.

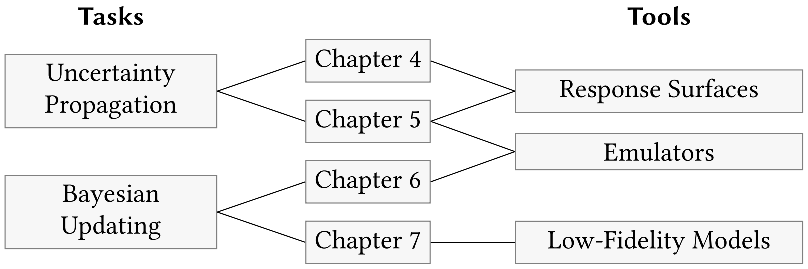

Figure 1.4.: Tasks and tools in the individual chapters

Contributions My own contributions to the overall goal of uncertainty quantification under time constraints are all presented in the second part. There are four contributions altogether, the first two focusing on uncertainty propagation and the last two on Bayesian updating.

- In Chapter 4, I have a look at polynomial response surface models, which are commonly used in stochastic collocation. Within this pre-existing methodology, I present an optimal sampling rule that is specifically adapted to the construction of polynomial response surfaces.
- In Chapter 5, I generalize the approach from the previous chapter and optimize both the sampling rule and the functional form of a response surface model. This chapter reveals an interesting connection between response surface models and emulators.
- In Chapter 6, I turn to Bayesian parameter inference problems and present a sequential sampling strategy designed for the solution of Bayesian inverse problems with emulators.
- Finally, in Chapter 7, I investigate what impact available measurement data have on the model selection between a reference model and a low-fidelity model.

The tasks and tools are linked to the four chapters as shown in Figure 1.4.

Epilogue The final part contains a summary, conclusions and an outlook on possible future work. Using my results from the second part, I discuss and evaluate the three hypotheses formulated above.

2. Tasks

In this chapter, I introduce the two basic tasks *uncertainty propagation* (see Section 2.1) and *Bayesian Updating* (see Section 2.2). For both tasks, I also provide a short discussion of quality measures. These are required to quantify how accurate the two tasks are performed when numerical methods are used.

In Figure 1.2 on page 8, I provided a schematic overview of the individual tasks.

2.1. Uncertainty Propagation

Given a random variable X of input parameters and a model function u, uncertainty propagation (UP) is the investigation of the stochastic properties of $Y := u(X)$. In other words: how does the uncertainty in X propagate through model u? The relationship between X, u and Y is schematically shown in Figure 2.1.

We might be interested in many different aspects of Y, such as the expected value or variance, the probability density function (pdf) or cumulative distribution function (cdf), exceedance probabilities or the sensitivity with respect to X.

In the following two sections, I will introduce the two protagonists of this thesis: the model parameters X (Section 2.1.1) and the model function u (Section 2.1.2). Afterwards, I will present the L^2-error as the appropriate error measure for UP problems (Section 2.1.3) and outline two well-known approaches to UP problems that are not applicable under the assumption of an expensive model function (Section 2.1.4).

2.1.1. Model Parameters

Throughout this work, model parameters are defined as those model inputs that are uncertain. Model inputs that are known with certainty are not

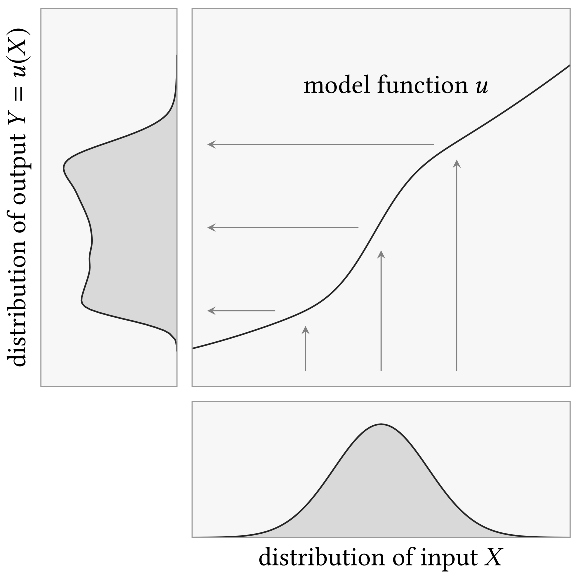

Figure 2.1.: Uncertainty propagation example

regarded as model parameters, even when they are called parameters in other works. Typical examples for model inputs that can be uncertain are material parameters, boundary conditions, initial conditions or geometric specifications. Location and time are usually regarded as system variables or coordinates, not as model parameters.

The uncertainty in the model parameters is described by a random variable (or random vector) X with values in the parameter domain $\Omega \subseteq \mathbb{R}^d$. According to the assumption of statistical uncertainty, see Section 1.1, we assume that the distribution of X is known, such that we can specify a pdf or a probability mass function. This is an important and very strong assumption because, in practice, a parameter may only be vaguely known in terms of expert knowledge or in form of a small number of measurement values. Such representations are less informative than a probability distribution and one would have to select an appropriate distribution first. This step is called uncertainty modeling. It is a big topic on its own, that falls out of the scope of this thesis.

These considerations raise one important questions: what is the meaning of the distribution of X? Under what condition is a distribution the *correct* distribution of a parameter? After all, the parameter is deterministic, only its value is not known precisely. This is a philosophical topic and I will only briefly comment on it. Throughout this thesis I adopt a Bayesian standpoint [9, 55]. That means that probabilities and probability distributions are not properties of the quantities under consideration, but properties of the knowledge or belief of the modeler. Therefore, different modelers may assign different distributions to the same quantity. This assignment is subjective. However, that does not mean that probability distributions are arbitrary. Distributions of different variables have to be consistent with the relation between the variables themselves. If, for example, variable X_1 is assumed to be uniformly distributed on the interval $[a, b]$, then the distribution of another variable $X_2 := X_1 + 1$ follows immediately as the uniform distribution on $[a + 1, b + 1]$.

In this perspective, uncertainty propagation does not claim to provide the correct distribution of some model output $Y = u(X)$ because there is no such thing as a *correct* distribution. Instead, it provides the distribution of Y that is consistent with the belief about X: if the modeler believes in a certain distribution of X and if she is consistent in her belief, then her belief about the distribution of Y follows directly and uniquely.

The model parameters I am using are assumed to be either continuous with a known pdf or discrete with a finite set of values. One operation that is frequently used is the expected value operator (or integration over the distribution of X) [158]. Let $f : \Omega \longrightarrow \mathbb{R}$ be a function. If X is a continuous random variable with pdf π, then the expected value of f is defined as the integral

$$\mathrm{E}\left[f(X)\right] = \int_\Omega f(x)\,\pi(x)\,\mathrm{d}x,$$

and if X is a discrete random variable, distributed on the values $x_1, \ldots, x_N$ with probabilities $w_1, \ldots, w_N$, then the expected value is defined as the sum

$$\mathrm{E}\left[f(X)\right] = \sum_{i=1}^{N} w_i f(x_i).$$

More generally, the expected value can then be written with a measure μ

describing the distribution of X:

$$\mathrm{E}\left[f\left(X\right)\right] := \int_{\Omega} f\left(x\right)\,\mathrm{d}\mu\left(x\right). \tag{2.1}$$

I will use this general formulation if the type of random variable (continuous or discrete) is not important. But as I will not use any results from measure theory, the notation with measure μ in Eq. (2.1) can just be thought of as a shortcut to mean one of the other two definitions of the expected value, and the term *measure* will just be used to describe the distribution of X.

I will stick to the usual notation in which random variables are denoted by upper case letters (X, Y, Z) and their possible values are denoted by lower case letters (x, y, z).

Statistical Dependence

Generally, the model parameters may be multivariate, having more than one component. Then, the variable X is a random vector and the measure μ describes the joint distribution of all components. To my knowledge, almost all publications in UQ that deal with more than one input parameter assume that the individual input parameters are independently distributed. In this work, I explicitly do not make this assumption at any point.

The independence assumption is very convenient. Many methods can be developed for the one-dimensional case and can then be generalized to the higher-dimensional case by simply applying the method to each dimension separately.

The classical justification of making the independence assumption is that possible dependencies between the model parameters can be removed by applying a suitable transformation, such as the Rosenblatt-transform [123] or the Nataf-transform [84, 90]. Imagine that the input parameter of the model is a random vector X and that the individual components have dependencies. Then, we can find a transform $\mathcal{T}$, such that $\hat{X} = \mathcal{T}^{-1}\left(X\right)$ is a random vector with independently distributed components. Now, the model function $u\left(\cdot\right)$ is replaced by the function $u\left(\mathcal{T}\left(\cdot\right)\right)$ and X is replaced by $\hat{X}$.

In my opinion, this approach is problematic. The transformation $\mathcal{T}$ introduces a coupling between the input distribution and the model function because the new function $u(\mathcal{T}(\cdot))$ contains information about the original parameter X. Conceptually, however, these are two separate questions: *"What values does the input parameter take?"* and *"How does the model react to a certain input?"* In a different context, Daniušis et al. formulated this idea as a postulate [38]: "If X causes Y, the distribution of X and the function f mapping X to Y are independent since they correspond to independent mechanisms of nature."

For two reasons, I believe it is advantageous to keep these two mechanisms separated.

1. Later in our analysis, we might learn something new about X and therefore change its distribution. The model function $u(\cdot)$ itself would remain as it is, while the function $u(\mathcal{T}(\cdot))$ would change according to the change in the transformation $\mathcal{T}$. If we had build a surrogate model of $u(\cdot)$, then it would still be useful. A surrogate model of $u(\mathcal{T}(\cdot))$, however, would become useless and we would have to build a new one.
2. Furthermore, it is reasonable to expect that the composition of two functions $u(\mathcal{T}(\cdot))$ would introduce new irregularities (e.g. nonlinear behavior) into the function instead of canceling irregularities. If we were to expect cancellation effects, then u and $\mathcal{T}$ would not be independent and this would violate the aforementioned postulate. For this reason, I expect the function $u(\cdot)$ to be easier to approximate by a surrogate than $u(\mathcal{T}(\cdot))$.

In summary, while an independence assumption is convenient, a transformation to independent variables brings new problems. I decided to avoid these problems by not making the independence assumption. All methods I developed naturally work for dependent parameters.

If the components of X are independent, then the corresponsing measure μ is called *separable.*

2.1.2. Model Function

The model function is a formal description of a simulation. It maps input parameters to model output:

$$u : \Omega \to \mathbb{R}.$$

In this notation, all simulation inputs that are known with certainty are assumed to be captured by the model function u already.

In practice, of course, the simulation output is not a scalar, but consists of multiple physical quantities that vary over space and/or time. If this is the case, then we regard the simulation output at each point in space and time and for each output quantity as an individual model function.

The reason for treating space and time different than the parameter domain is that, in most simulations, the equations are coupled in space and time, but are not coupled with respect to the parameters. It is possible to obtain the model output for a specific value of the input parameters without considering other parameter values, but it is usually not possible to calculate the model output at one point in the spatial or temporal domain without calculating it in all other points as well.

As formulated in Sections 1.2, the model function is assumed to be computationally expensive.

Furthermore, we assume that the model function is square integrable with respect to the measure μ. The space of all square integrable functions on Ω with respect to μ is defined as

$$L^2 := \left\{ f : \Omega \to \mathbb{R} \middle| \int_\Omega f(x)^2 \, \mathrm{d}\mu(x) < \infty \right\}.$$

In full length, this function space would be called $L^2(\mu)$, but as we are only considering one L^2-space, we will simply call it L^2. It is a Hilbert space, equipped with an inner product [33]

$$\langle f, g \rangle_{L^2} = \int_\Omega f(x) g(x) \, \mathrm{d}\mu(x), \tag{2.2}$$

which induces a norm

$$\|f\|_{L^2} = \sqrt{\langle f, f \rangle_{L^2}}.$$

Again for brevity, we will omit the indices in the inner product and norm and simply write $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$, as long as no confusion is to be expected. Two functions f, g are called orthogonal if $\langle f, g \rangle = 0$.

In short, we assume that $u \in L^2$. From the definition of L^2, it directly follows that the model output $u(X)$ has a finite variance.

Unfortunately, in one aspect, the space L^2 is not a good description for model functions: in L^2, two functions f and g are regarded equal if $\|f - g\| = 0$. This condition, however, does not imply, that $f(x) = g(x)$ for all $x \in \Omega$ because deviations in single points do not change the value of the integral.

For this reason, functions that cannot be distinguished via the L^2-norm are summarized in equivalence classes, so that the space L^2 becomes a space of equivalence classes. The problem with these equivalence classes is that point evaluations are not defined on them. For example, if we know that a function f lies in the same equivalence class as the null-function, i. e., $\int_\Omega f^2(x)\, \mathrm{d}\mu(x) = 0$, then it does not necessarily follow, that $f(x) = 0$ for all $x \in \Omega$. The function f might even take arbitrary values in individual points in Ω. Therefore, the value of $f(x)$ for individual points $x \in \Omega$ is undefined as long as we only know the equivalence class in which f lies.

The fact that point evaluations are not defined on L^2 clearly contradicts the idea of describing a simulation by a model function: the model function u is defined by the input-output relationship of a black-box simulation in the first place so it naturally can be evaluated in a pointwise manner. To avoid this contradiction, I will assume that the model function u lies in a space $\mathbb{U} \subset L^2$ on which point evaluation is defined. For example, if u is known to be continuous, then we can define $\mathbb{U}$ as the space of all continuous functions in L^2. Of course, the model function does not necessarily have to be continuous, so other definitions of $\mathbb{U}$ are possible, too. As we will see in the later chapters, each method requires additional assumptions about the model function anyways, so, at this point, we do not need to define the space $\mathbb{U}$ more specifically.

To summarize, we assume that the model function lies in a space $\mathbb{U}$ on which point evaluation is defined and in which we can use the same norm and inner product as in L^2.

2.1.3. Quality Measure: L^2-Error

As noted before, uncertainty propagation is the investigation of the stochastic properties of $Y := u(X)$. As Y is a random variable due to its definition via X, we are interested in all properties it has as a random variable. This means that finding the distribution of Y in terms of its pdf is not enough. The pdf alone does not contain information about the dependency between Y and other variables. Therefore, in UP problems, we have to understand (and also approximate) Y as a function of X.

Using a numerical method, we obtain an approximation $\tilde{Y}$ of Y. Similar to Y itself, also $\tilde{Y}$ is a function of X, so we write $\tilde{Y} = \tilde{u}(X)$. The natural measure for the accuracy of $\tilde{Y}$ is the L^2-norm error of the difference between Y and $\tilde{Y}$, which is explained by the L^2-norm error of the corresponding functions

u and $\tilde{u}$, see Section 2.1.2:

$$\left\|Y - \tilde{Y}\right\| = \|u - \tilde{u}\| = \int_{\Omega} \left(u\left(x\right) - \tilde{u}\left(x\right)\right)^2 \,\mathrm{d}\mu\left(x\right).$$

The L^2-norm error is a general-purpose error type. It only becomes zero if $Y = \tilde{Y}$, so an error of zero guarantees that the probability of Y and $\tilde{Y}$ taking two different values is zero. Furthermore, if the L^2-norm error is small, then also all other possible error measures such as error in the pdf, in expected value or in variance, become small as well. The converse is not true: two random variables might have the same expected value, variance or even pdf but that does not guarantee that the two random variables are the same. For example, if Y is uniformly distributed on $[0, 1]$, then $Y' = 1 - Y$ is also uniformly distributed on $[0, 1]$, but clearly Y and Y' are different random variables. Therefore, error measures other than the L^2-norm error should only be used if there is a special reason for doing so (e. g. if the expected value is the only quantity of interest). If we have not decided on a specific quantity of interest, then we should use the L^2-norm error.

The L^2-norm error has the additional advantage that it is induced by the inner product. Therefore, we can build methods on the concept of orthogonality.

2.1.4. Analytical Approach and the Monte-Carlo Method

Let me briefly outline two approaches to uncertainty propagation that can generally solve uncertainty propagation problems, but that are impractical under the assumption of an expensive black-box model function.

First, by the definition of the output Y, we can derive an expression for its pdf π_Y via the so-called cdf method [158]. Assume that the input X has a known pdf π_X. The pdf of Y is the derivative of its cdf F_Y: $\pi_Y\left(y\right) = F_Y'\left(y\right)$, where the cdf is defined as $F_Y\left(y\right) := P\left[Y < y\right]$. Now we define the set $A\left(y\right) := \{x \in \Omega | u\left(x\right) < y\}$. The probability of $Y < y$ is the same as the probability of $x \in A\left(y\right)$, so we find

$$F_Y\left(y\right) = \int_{A(y)} \pi_X\left(x\right)\,\mathrm{d}x,$$

and obtain the expression

$$\pi_Y(y) = \frac{\partial}{\partial y} \int_{A(y)} \pi_X(x)\, \mathrm{d}x.$$

If both X and Y have scalar values and u is invertible, this expression can further be simplified with the transformation method [158]. We define the inverse function of u as $v := u^{-1}$ and denote its derivative by v'. Then the pdf of Y is

$$\pi_Y(y) = \pi_X(v(y)) \cdot |v'(y)|$$

Both the cdf method and the transformation method are not of any practical use if the model function u is an expensive black-box function: we do not have an explicit expression for u, so we cannot invert or derive it.

Second, to calculate statistics of any function of X, the Monte-Carlo (MC) method can be used. If we draw an independent and identically distributed (i.i.d.) sample of the model parameters $x_1, \ldots, x_N \sim X$, then $f(x_1), \ldots, f(x_N)$ is an i.i.d. sample of $f(X)$. We can estimate the expected value of $f(X)$ via [89]

$$\mathrm{E}[f(X)] \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i). \tag{2.3}$$

The error between the expected value and its estimate is called *stochastic error.* With increasing sample size, it converges to zero at a rate of $1/\sqrt{N}$ [89]. This can be proved by calculating the standard deviation of the estimate. Let $Y_1, \ldots, Y_N$ be i.i.d. random variables with a finite variance. It follows

$$\begin{aligned} \mathrm{Var}\left[\frac{1}{N}\sum_{i=1}^{N} Y_i\right] &= \frac{1}{N^2}\mathrm{Var}\left[\sum_{i=1}^{N} Y_i\right] \\ &= \frac{1}{N^2}\sum_{i=1}^{N} \mathrm{Var}[Y_i] \\ &= \frac{1}{N}\mathrm{Var}[Y_1] \end{aligned}$$

The standard deviation, being the square root of the variance, then decays at a rate of $1/\sqrt{N}$. This convergence rate is independent of the dimension of the input domain and the MC method only requires the assumption of finite variance. If the model function is computationally expensive, however, then the Monte Carlo convergence rate can be regarded as too slow:

for each digit of accuracy in the estimate, the sample size has to be increased by a factor of 100.

Because of its wide applicability, we can regard the MC method as a base line method: any other method can only be useful if it is faster than the MC method.

2.2. Bayesian Updating

Bayesian updating (BU), also called Bayesian inference, is the process of using measurement data or observations to update the knowledge about unobserved quantities. An observation of the *input* parameters of a model does not fall into this category because such observations can be handled with uncertainty propagation methods presented in the previous section. The interesting case is the assimilation of observations about model output quantities. I differentiate between two different types of BU problems: a measurement of an output quantity may give us information about (1) the input parameters or (2) other quantities of interest.

The first type of problem is called "(Bayesian) inverse problem" or "parameter inference problem" [142]. To my knowledge, the second type of problem does not have an own name, so I refer to it as "general Bayesian updating". The different flow of information in these two variants of Bayesian updating is schematically shown in Figure 1.2 on page 8. General BU problems can be regarded as the composition of, first, an inverse problem (from the measurement to the input) and, second, an uncertainty propagation problem (from the input to the quantity of interest) and it is possible to solve them using this order.

The solution to an inverse problem is provided by Bayes' theorem. In a general form, it is presented for example by Stuart [137]. I will restrict myself to two specific types of random variables: continuous random variables and discrete random variables. For these two types of variables, Bayes' theorem takes different forms, which are presented in Sections 2.2.1 and 2.2.2. As a quality measure for numerical solutions of BU problem, I propose the use of statistical distances. The two distances used in this work are presented in Section 2.2.3.

2.2.1. Bayes' Theorem for Continuous Input Parameters

Assume that the input parameters X have a pdf π. We describe the observable quantity by a random variable Z and assume a functional relation between X and Z of the form $Z = f(X, \varepsilon)$, where ε denotes additional noise. Now Z is observed as $Z = z$ and we are interested in finding the posterior pdf $\hat{\pi}$ of X which is defined as the pdf of X conditional to the event $Z = z$, i. e., $\hat{\pi}(x) = \pi_{X|Z}(x|z)$.

Bayes' theorem provides a formula for the posterior distribution $\hat{\pi}$:

$$\hat{\pi}(x) = \frac{L(x)\,\pi(x)}{\int_\Omega L(x')\,\pi(x')\,\mathrm{d}x'}.$$

Here, L denotes the likelihood function. Before I provide a precise definition of the likelihood L, it is worth noting that the normalizing term in the denominator $\int_\Omega L(x')\,\pi(x')\,\mathrm{d}x'$ is independent of x and can be regarded as a constant coefficient. Many numerical methods do not require the calculation of this coefficient, so for brevity, we write

$$\hat{\pi}(x) \propto L(x)\,\pi(x)\,.$$

In short, Bayes' theorem scales the probability mass proportionally to the likelihood.

The definition of the likelihood function depends on what type of random variable Z is. If Z is a discrete random variable, then the likelihood is defined as the conditional *probability* of observing $Z = z$ under the condition that $X = x$,

$$L(x) = P[Z = z|X = x]\,, \tag{2.4}$$

and if Z is a continuous random variable, the likelihood is the conditional probability *density* of the same event:

$$L(x) = \pi_{Z|X}(z|x)\,. \tag{2.5}$$

The specific functional form of L depends on the measurement model. The most commonly used measurement model assumes an independent, additive measurement error

$$Z = u(X) + \varepsilon, \tag{2.6}$$

where ε is a random error with known pdf π_ε. As discussed in Section 1.1, the model function u itself does not contain a model error, so the difference between Z and $u(X)$ does only consist of measurement errors.

According to this measurement model, the likelihood reads

$$L(x) = \pi_{Z|X}(z|x) = \pi_\varepsilon(z - u(x)).$$

The computational bottleneck in calculating a posterior distribution $\hat{\pi}$ lies in the likelihood term. For each evaluation of $L(x)$, the model response $u(x)$ has to be evaluated once.

Figure 2.2 exemplarily shows the relation between model function, data, prior, likelihood and posterior. The x-axis is the parameter axis. The top plot shows the model function u and the measured data value $z \pm$ one standard deviation of the measurement error ε. In the bottom plot, the prior, the posterior and the likelihood function are shown. The peak of the likelihood is at the intersection of the model function with the data. The posterior is the product of prior and likelihood, expressing a combination of the prior knowledge and the fit between model and data. Therefore, its peak typically lies somewhere between the peaks of the prior and of the likelihood.

2.2.2. Bayes' Theorem for Discrete Input Parameters

If the input parameters X are discretely distributed with values $x_1, \dots, x_N$ and weights $w_1, \dots, w_N$, Bayes' theorem takes a slightly different form. The basic idea, however, is the same: the probability mass is scaled according to the likelihood. The posterior weights $\hat{w}_i$ follow as

$$\hat{w}_i = \frac{L_i w_i}{\sum_{j=1}^{N} L_j w_j}, \tag{2.7}$$

or in short

$$\hat{w}_i \propto L_i w_i.$$

With the additive measurement model from Eq. (2.6), the likelihood reads

$$L_i = \pi_\varepsilon(z - u(x_i)).$$

For details on this formulation, see Smith and Gelfand [135].

In the setting of discrete input parameters, the solution of general Bayesian updating problems becomes particularly simple. Let us assume that the

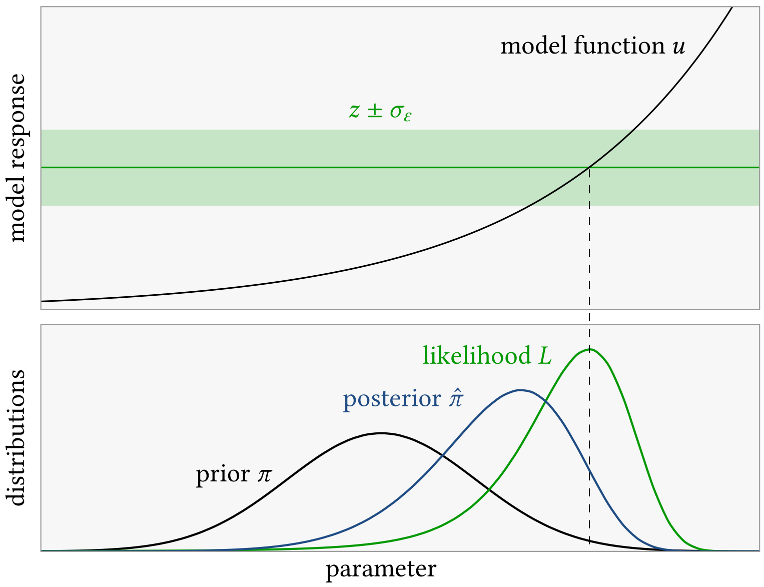

Figure 2.2.: Bayesian inverse problem example

simulation returns both the observable quantity $u(x)$ and a quantity of interest $\mathrm{QoI}(x)$. After running the simulation for all possible parameter values, we obtain a complete collection of both quantities: $u(x_1), \ldots, u(x_N)$ and $\mathrm{QoI}(x_1), \ldots \mathrm{QoI}(x_N)$. Using Bayes' theorem and the measurable quantity, we can compute the posterior weights $\hat{w}_i$ via Eq. (2.7). If we attach these posterior weights realization-wise to the samples of $\mathrm{QoI}(x_i)$, we get the posterior distribution of the quantity of interest in form of a weighted sample.

The discrete formulation of Bayes' theorem is often used to draw a sample from the posterior distribution of X, even if X itself is a continuous random variable. Instead of calculating the posterior pdf of X and drawing a sample from this pdf, it is often much easier to draw a prior sample of X and assign each realization a weight according to Eq. (2.7). One can also obtain an unweighted sample by means of rejection sampling [89].

2.2.3. Quality Measure: Statistical Distances

The target quantity in Bayesian updating problems is the posterior distribution. To compare different numerical methods in their ability to calculate a posterior distribution, we need a measure for the distance between distributions, so called statistical distances. The L^2-norm error used in uncertainty propagation cannot be used as a quality measure in Bayesian updating because we are interested in the distance between different measures and not between different random variables on the same measure. The two statistical distances used in this work are the Kullback-Leibler divergence and the Earth Mover's Distance. Other distance measures exist (e.g. Hellinger distance [149], total variation distance [124]), but are not further considered here.

The Kullback-Leibler divergence (KL divergence) can be interpreted as the loss of information if a pdf π is replaced by a different pdf $\tilde{\pi}$ [79]. It is defined as

$$D_{\mathrm{KL}}\left(\pi \| \tilde{\pi}\right) := \int_{\Omega} \pi\left(x\right) \cdot \ln \frac{\pi\left(x\right)}{\tilde{\pi}\left(x\right)} \,\mathrm{d}x.$$

The KL divergence is not a metric in the strict sense, because it is not symmetric: $D_{\mathrm{KL}}\left(\pi \| \tilde{\pi}\right) \neq D_{\mathrm{KL}}\left(\tilde{\pi} \| \pi\right)$. For that reason, it is called a divergence. Usually the order of the two pdfs π and $\tilde{\pi}$ is chosen such that π is the reference or "correct" pdf and $\tilde{\pi}$ is its approximation.

The Earth Mover's distance (EMD) has a very intuitive meaning: if a distribution is thought of as a pile of earth, then the EMD between two distributions is the minimal work required to turn one distribution into the other one [125]. In higher dimensions, the EMD is not easily computed, but for one-dimensional distributions, the EMD between the distributions of two random variables X_1 and X_2 is simply the area between their cdfs F_{X_1} and F_{X_2} [28]:

$$D_{\mathrm{EM}}\left(F_{X_1}, F_{X_2}\right) = \int_{\Omega} \left| F_{X_1}\left(x\right) - F_{X_2}\left(x\right) \right| \mathrm{d}x.$$

The EMD has the advantage that it does not require the existence of a pdf. If two random variables are discretely distributed (i.e. they are given in terms of weighted samples), then their EMD can be calculated analytically (because the cdf functions are step functions), while the KL divergence would require a pdf estimate first, which potentially introduces additional errors. In those applications where I handle pdfs anyway, I will compare solutions in terms of their KL divergence. Otherwise, I will use the EMD.

3. Tools

To perform the two tasks of uncertainty propagation and Bayesian updating presented in Chapter 2, I use three main tools:

1. response surfaces,
2. emulators,
3. low-fidelity models.

Recall that the purpose of all these methods is to find a function that behaves similar to the model function, but that is faster to evaluate. This replacement function is called the "surrogate model".

In the introductory section 1.3, I already gave an overview of these tools in terms of their practical differences. In this section, I will present the tools from a mathematical point of view and introduce my notation.

3.1. Response Surfaces

A response surface is a surrogate model in form of a mathematical function fitted to the response of the model function. Once such a surrogate model is constructed, all subsequent calculations can be accelerated by simply replacing the model function by its surrogate. We assume that the evaluation of a response surface is so fast, that the computational costs of those subsequent calculations can be neglected. Therefore, the overall costs only consist of the construction of the response surface, as this step requires evaluations of the model function, see Section 1.2. A response surface is constructed from input-output pairs of the model function which we regard as a black-box. Therefore, response surface methods are nonintrusive.

In my work, I restrict myself to linear response surface methods. A method is linear if the operator that maps a model function to a repose surface is a linear operator. This does not mean that the response surface itself is a linear function. Response surface methods become non-linear if the ansatz functions are chosen adaptively, such as in sparse PCE methods [74, 170].

By restricting ourselves to linear methods, we exclude adaptive methods of this kind.

In the two following sections, I first formulate a very general representation of linear, nonintrusive response surface methods, and then I present the stochastic collocation (SC) method. SC is one of the most widely used response surface methods in UQ [5, 47, 48, 97, 166, 168], and it is based on polynomial chaos expansions [57, 160, 167, 169].

3.1.1. General Formulation

As before, we assume that the model function lies in the space $\mathbb{U}$. A response surface method can now be described as an operator $\mathcal{P} : \mathbb{U} \to \mathbb{U}$ that maps a model function u to its surrogate $\tilde{u} = \mathcal{P}u$. Being a nonintrusive method, the operator $\mathcal{P}$ internally consists of two steps: first, the evaluation of the model function in a finite number of points, and second, the construction of the response surface from the model responses. Formally, I suggest to write this operator in the following form:

$$\mathcal{P} = \Phi\mathcal{E}_{\boldsymbol{x}}. \tag{3.1}$$

Here, $\mathcal{E}_{\boldsymbol{x}}$ denotes the evaluation operator that evaluates a function on a given set of points. For an n-tuple of points $\boldsymbol{x} = (x_1, \ldots, x_n) \in \Omega^n$ it is defined as

$$\mathcal{E}_{\boldsymbol{x}} : \mathbb{U} \to \mathbb{R}^n : f \mapsto [a_1, \ldots, a_n]^\top = [f(x_1), \ldots, f(x_n)]^\top .$$

Throughout this thesis, I will call the points at which the model function is evaluated *nodes*.

The operator Φ maps a vector of n model responses to a linear combination of functions $\Phi_1, \ldots, \Phi_n \in \mathbb{U}$. It is defined as

$$\Phi : \mathbb{R}^n \to \mathbb{U} : [a_1, \ldots, a_n]^\top \mapsto \sum_{i=1}^{n} a_i\Phi_i.$$

The functions Φ_i can be thought of as unit response functions: if the evaluation operator $\mathcal{E}_{\boldsymbol{x}}$ returns a unit vector (1 in one component, 0 everywhere else), then the operator Φ returns one of the unit response functions. In loose notation, the operator Φ can be understood as a row vector containing the unit response functions $\Phi = [\Phi_1, \ldots, \Phi_n]$. The image of $\mathcal{P}$ is the span of the unit response functions $\mathbb{P} := \operatorname{span}\{\Phi_1, \ldots, \Phi_n\}$ and we call this space the *ansatz space*. Furthermore, we denote the space of all operators of the form in Eq. (3.1) with n nodes as $\mathfrak{P}_n$.

Any linear, nonintrusive response surface method (including interpolation and least-squares fit) can be represented in the form given in Eq. (3.1). Even though I did a literature review, I did not see this kind of operator representation elsewhere in the literature. Given that it is very much straightforward, however, I would expect that similar formulations already exists in other fields of mathematics or engineering.

3.1.2. Stochastic Collocation

We now turn to the perhaps most widely used response surface method in UQ: stochastic collocation based on polynomial chaos expansions.

The Polynomial Chaos Expansion and its Properties

Under some assumption about the distribution of X, the space of all polynomials is a dense subspace of L^2 [49]. If this is the case, then any model function $u \in L^2$ can be represented by its polynomial chaos expansion (PCE) [57, 160, 167, 169], which has this form:

$$u = \sum_{i=1}^{\infty} c_i \Psi_i. \tag{3.2}$$

The infinite sum converges in the L^2-sense. In this expansion, the c_i are expansion coefficients and the Ψ_i are polynomials that form an orthonormal basis (ONB) of L^2, i.e. any two polynomials Ψ_i and Ψ_j satisfy

$$\left\langle \Psi_i, \Psi_j \right\rangle = \delta_{ij}, \tag{3.3}$$

where δ is the Kronecker delta. With this property, a PCE is a generalized Fourier series expansion (not with respect to a trigonometric basis, but with respect to an ONB of polynomials) [167].

Typically, the polynomials satisfy additional conditions. The first polynomial is usually the constant $\Psi_1(x) = 1$. Then, each following polynomial Ψ_i contains one additional monomial that is not contained in the previous polynomials $\Psi_1, \ldots, \Psi_{i-1}$. Very often, the polynomials are ordered by (total) degree.

From the orthonormality, we can derive a simple expression for the expected value of the basis functions. The first polynomial is the constant 1, so we find

$$\begin{aligned} \mathrm{E}\left[\Psi_i\left(X\right)\right] &= \int_{\Omega} \Psi_i \cdot 1 \,\mathrm{d}\mu\left(x\right) \\ &= \langle \Psi_i, \Psi_1 \rangle = \delta_{i1}. \end{aligned} \tag{3.4}$$

If X is one-dimensional and the distribution of X is a very common one (such as uniform or normal distribution), then the corresponding orthogonal polynomials typically have their own name (such as Legendre polynomials or Hermite polynomials) [2]. The ortho*normal* polynomials we are using here differ from these only in a normalizing factor. If the distribution of X is not a common one, then orthonormal polynomials can still be constructed using orthonormalization algorithms. It is possible orthogonalize via the Gram-Schmidt process [163], by solving a linear system [112] or by using a three-term-recurrence relation [53]. I recommend to use the latter because it is numerically more stable than the other two.

If X is higher-dimensional with independent components, then the multivariate orthonormal polynomials can be constructed by simply multiplying the individual univariate polynomials for each dimension [81]. The resulting multi-variate polynomials will automatically be orthonormal. However, if the components of X are dependently distributed, see Section 2.1.1, then it is not enough to simply combine the individual dimensions. Instead, an additional orthogonalization step has to be done.

The expansion given in Eq. (3.2) has a number of convenient properties. Thanks to the orthonormality of the basis polynomials, there is an analytical expression for the expansion coefficients (which is typical for Fourier expansions [167])

$$c_i = \langle u, \Psi_i \rangle\,. \tag{3.5}$$

This expression, however, is of little direct practical use, as the inner product involves integration of u over Ω, see Eq. (2.2).

A PCE becomes a response surface if it is truncated after p terms:

$$u^{\star} = \sum_{i=1}^{p} c_i \Psi_i. \tag{3.6}$$

This kind of response surface is appealing because the part of the series that has been cut off is orthogonal to $u^{\star}$. Therefore, the surrogate $u^{\star}$ is the orthogonal projection of u onto the ansatz space $\mathbb{P} = \operatorname{span}\{\Psi_1, \dots, \Psi_n\}$, and so $u^\star$ is the best approximation of u in $\mathbb{P}$ [167]. In other words,

$$\left\| u - u^\star \right\| = \inf_{f \in \mathbb{P}} \| u - f \| . \tag{3.7}$$

The error between the model function u and its best approximation $u^\star$ is called *truncation error*.

In practice, calculating the coefficients $c_0, \dots, c_n$ can only be done approximately, which leads to an approximated polynomial

$$\tilde{u} = \sum_{i=1}^{p} \tilde{c}_i \Psi_i . \tag{3.8}$$

The error between the best approximation $u^\star$ and the actual surrogate $\tilde{u}$ is called *approximation error*.

The overall (or total) error between u and $\tilde{u}$ can be split into the truncation error and approximation error:

$$\begin{aligned} \underbrace{u - \tilde{u}}_{\text{total error}} &= \underbrace{u - u^\star}_{\text{truncation error}} + \underbrace{u^\star - \tilde{u}}_{\text{approximation error}} \\ &= \sum_{i=p+1}^{\infty} c_i \Psi_i + \sum_{i=1}^{p} \left(c_i - \tilde{c}_i \right) \Psi_i . \end{aligned}$$

Due to the orthonormality of the basis functions Ψ_i, the two errors are orthogonal and their squared L^2-norms add up

$$\begin{aligned} \| u - \tilde{u} \|^2 &= \left\| u - u^\star \right\|^2 + \left\| u^\star - \tilde{u} \right\|^2 \\ &= \sum_{i=p+1}^{\infty} c_i^2 + \sum_{i=1}^{p} \left(c_i - \tilde{c}_i \right)^2 . \end{aligned} \tag{3.9}$$

Another convenient property of a PCE is that the expected value and variance of $\tilde{u}$ can be calculated directly from the coefficients:

$$\begin{aligned} \mathrm{E}_X \left[\tilde{u}\left(X \right) \right] &= \tilde{c}_0, \\ \mathrm{Var}_X \left[\tilde{u}\left(X \right) \right] &= \sum_{i=2}^{p} \tilde{c}_i^2 . \end{aligned}$$

Tensor-Grid Space

	x_1^0	x_1^1	x_1^2	x_1^3	x_1^4
x_2^0	●	●	●	●	●
x_2^1	●	●	●	●	●
x_2^2	●	●	●	●	●
x_2^3	●	●	●	●	●
x_2^4	●	●	●	●	●

Total-Degree Space

	x_1^0	x_1^1	x_1^2	x_1^3	x_1^4
x_2^0	●	●	●	●	●
x_2^1	●	●	●	●	
x_2^2	●	●	●		
x_2^3	●	●			
x_2^4	●				

Figure 3.1.: Comparison of a tensor-grid space and total-degree space of polynomials up to degree 4 in two dimensions. Each circle represents one basis polynomial Ψ_i included in the space.

Ansatz Spaces in Higher-Dimensional Problems

In one-dimensional problems, the choice of the ansatz spaces is restricted to the choice of the polynomial degree. In higher-dimensional problems, the modeler has some freedom in choosing the ansatz space because of the many different cross terms between the individual dimensions. The two most common choices are tensor grid spaces and total degree spaces. In a tensor grid space, each dimension is assigned a degree and the tensor grid space consists of all possible cross terms up to these degrees. A total degree space consists of all polynomials up to a maximum total degree. A schematic comparison of these two types of spaces is shown in Figure 3.1.

The ansatz space does not necessarily have to be chosen by the modeler. There are also sparse, adaptive methods that select the ansatz space depending on the model response [14, 15]. These methods are not linear and therefore not considered here.

Calculating the Expansion Coefficients

The main difficulty in constructing a PCE lies in the calculation of the expansion coefficients c_i. Since the model function is assumed to be a black-box function, we can only use nonintrusive methods for calculating these coefficients. There exist also intrusive methods, such as the stochastic Galerkin method [81, 167], but I will not go into details about such methods. Nonintrusive PCE methods are also called stochastic collocation [67, 168]. The two most commonly used stochastic collocation methods are direct calculation by a quadrature rule and calculation by least-squares approximation. I refer to these two approaches as the *quadrature approach* and the *least-squares approach*, respectively. Both approaches rely on quadrature rules. A quadrature rules consists of nodes $x_1, \ldots, x_n \in \Omega$ and weights $w_1, \ldots, w_n \in \mathbb{R}$ such that an integral can be approximated by a sum [61]:

$$\int_\Omega f(x)\, \mathrm{d}\mu(x) \approx \sum_{j=1}^{n} w_j f\left(x_j\right). \tag{3.10}$$

Quadrature Approach The first approach starts with the analytical expression for the coefficients $c_i = \langle u, \Psi_i \rangle$ [Eq. (3.5)], and inserts a quadrature rule directly [58, 82, 121]. We obtain

$$\tilde{c}_i = \sum_{j=1}^{n} w_j u\left(x_j\right) \Psi_i\left(x_j\right). \tag{3.11}$$

In the literature, this approach is also called "nonintrusive spectral projection" (NISP) [121] or "discrete expansion"[131].

Least-Squares Approach The second approach uses the fact that the exact polynomial $u^\star$ minimizes the L^2-norm error between u and $u^\star$, see Eq. (3.7). The L^2-norm contains an integral and inserting a quadrature rule here, we obtain a least-squares problem [25, 66, 70, 138, 165]:

$$\tilde{u} = \underset{f \in \mathbb{P}}{\operatorname{argmin}} \sum_{j=1}^{n} w_j \left[u\left(x_j\right) - f\left(x_j\right)\right]^2. \tag{3.12}$$

Inserting the approximate polynomial [Eq. (3.8)], we can obtain the coefficients as the solution of a linear least-squares system which can be solved via normal equations or a pseudo-inverse. Furthermore, if the number of nodes n is equal to the number of basis functions n, then the least-squares approximation becomes an interpolation. Even though I introduced the least-squares approach as an application of quadrature rules, it can also be understood as a least-squares functional approximation without the notion of quadrature rules. Therefore, I will refer to the method of choosing nodes and weights as a *sampling rule*, rather than a quadrature rule.

In the literature, the quadrature approach [121, 166] and the least-squares approach [11, 23, 44, 68] are mostly treated separately and I am not aware of any work attempting to compare these two approaches. In my view, both approaches are special cases of a more general formulation. Note that both the quadrature approach and the least-squares approach result in a linear relationship between the model responses $u\left(x_j\right)$ and the coefficients $\tilde{c}_i$. For this reason, I propose the following formulation:

$$\left[\tilde{c}_1, \ldots, \tilde{c}_p\right]^\top = A\left[u\left(x_1\right), \ldots, u\left(x_n\right)\right]^\top,$$

where A is a $p \times n$ matrix encoding the calculation rule of the coefficients. This formulation includes all possible linear relationships between model responses and coefficients. We obtain the quadrature approach by selecting $A_{ij} = w_j \Psi_i\left(x_j\right)$ (compare with Eq. (3.11)), and the least-squares approach by solving Eq. (3.12) via the normal equations.

This general approach can be expressed with the following operator

$$\mathcal{P} = \Psi A \mathcal{E}_{\boldsymbol{x}}. \tag{3.13}$$

I am not aware of any other scientific work that uses an operator representation like this.

The operator Ψ maps a vector of coefficients to a linear combination of the basis functions Ψ_i ($\Psi\left[c_1, \ldots, c_p\right] = \sum_{i=1}^{p} c_i \Psi_i$) and, as before, $\mathcal{E}_{\boldsymbol{x}}$ is the evaluation operator. This general approach has more degrees of freedom than the quadrature approach or least-squares approach: matrix A has np entries, while the other two approaches have n weights each.

This formulation is very similar to the general formulation $\mathcal{P} = \Phi \mathcal{E}_{\boldsymbol{x}}$ provided earlier in Eq. (3.1). The main difference is that we regard the operator Ψ as constant and given, so that we can only change A and $\boldsymbol{x}$, while in the general formulation, the operator Φ can freely be changed. So, for a fixed Ψ, the space of all operators $\Psi A \mathcal{E}_{\boldsymbol{x}}$ is a subspace of the operators of the form $\Phi \mathcal{E}_{\boldsymbol{x}}$.

The accuracy of the expansion coefficients relies heavily on the choice of an appropriate quadrature or sampling rule (i.e. the choice of $\boldsymbol{x}$ and w or $\boldsymbol{x}$ and A). In the one-dimensional case, Gaussian quadrature rules are regarded optimal (with respect to the polynomial order) [139, 151]. These, however, can not be easily generalized to higher-dimensional cases [61, 71]. A generalization to higher-dimensional cases is only possible if the components of X are independently distributed. As stated in Section 2.1.1, we generally do not assume that this is the case. But even in those cases where the components are independently distributed, the higher-dimensional generalizations of Gaussian quadrature rules have a downside: The number of nodes is the product of the numbers of nodes for each individual dimension. This means that the number of nodes cannot be chosen freely and, therefore, the quadrature rules are not flexible. As I mentioned ealier, it is desirable to use flexible methods. For these reasons, one of my contributions is devoted to the selection of nodes and weights via optimization, see Chapter 4.

3.1.3. The Lebesgue Constant and Operator Norms

As we saw in the previous sections, a response surface method can be expressed as an operator $\mathcal{P}$ on the space $\mathbb{U}$. In this section, I introduce two possible ways of measuring how good an operator is as a response surface method: the Lebesgue constant and operator norms.

The Lebesgue Constant

The Lebesgue constant is a common measure for the accuracy of interpolation methods, but it can equally well be used for response surface methods other than interpolation [148]. Let $\mathcal{P}$ be an operator that maps a function u to a response surface $\mathcal{P}u$, and define the ansatz space as $\mathbb{P} := \operatorname{Im}(\mathcal{P})$. As before, we define the best approximation of u in $\mathbb{P}$ as $u^\star := \operatorname{argmin}_{f\in\mathbb{P}} \|u - f\|$ and we assume that u is known to lie in a space $\mathbb{U}$. The Lebesgue constant relates the achieved error (the total error) with the smallest possible error in $\mathbb{P}$ (the truncation error). The Lebesgue constant of the operator $\mathcal{P}$ is the smallest number $\Lambda(\mathcal{P})$, so that for any model $u \in \mathbb{U}$ it holds [122, 148]

$$\|u - \mathcal{P}u\| \leq (1 + \Lambda(\mathcal{P})) \left\|u - u^\star\right\|. \tag{3.14}$$

Eq. (3.14) has to hold for any model $u \in \mathbb{U}$, so the Lebesgue constant satisfies

$$\Lambda(\mathcal{P}) = \sup_{u\in\mathbb{U}} \frac{\|u - \mathcal{P}u\|}{\|u - u^\star\|} - 1. \tag{3.15}$$

In the original definition of the Lebesgue constant, the norm $\|\cdot\|$ in Eq. (3.14) is the infinity norm $\|f\|_\infty := \sup\{|f(x)| : x \in \Omega\}$, but the concept of comparing the total error with the truncation error is also useful for other definitions of the norm. I will use the L^2-norm, but still call $\Lambda(\mathcal{P})$ the "Lebesgue constant".

Operator Norms

Closely related to Lebesgue constants are *operator norms*. Let $\mathbb{V}$ and $\mathbb{W}$ be two normed vector spaces with their norms $\|\cdot\|_{\mathbb{V}}$ and $\|\cdot\|_{\mathbb{W}}$. The space of all bounded linear operators from $\mathbb{V}$ to $\mathbb{W}$ is denoted by $\mathfrak{L}(\mathbb{V}, \mathbb{W})$. For an operator $\mathcal{A} \in \mathfrak{L}(\mathbb{V}, \mathbb{W})$, the operator norm is defined as [33]

$$\|\mathcal{A}\|_{\mathfrak{L}(\mathbb{V},\mathbb{W})} = \sup_{f \in \mathbb{V}\setminus\{0\}} \frac{\|\mathcal{A}f\|_{\mathbb{W}}}{\|f\|_{\mathbb{V}}}. \tag{3.16}$$

From this definition, it follows immediately, that operator norms are submultiplicative: for any element $f \in \mathbb{V}$ the following inequality holds

$$\|\mathcal{A}f\|_{\mathbb{W}} \leq \|\mathcal{A}\|_{\mathfrak{L}(\mathbb{V},\mathbb{W})} \|f\|_{\mathbb{V}}. \tag{3.17}$$

An operator is called bounded if it has a finite operator norm. To show that an operator is not bounded, it is enough to find a sequence of elements $f_1, f_2, \dots \in \mathbb{V}$ such that $\lim_{i\to\infty} \|f_i\|_{\mathbb{V}} = 0$, but $\lim_{i\to\infty} \|\mathcal{A}f_i\|_{\mathbb{W}} \neq 0$. This can only happen if $\mathbb{V}$ is infinite dimensional.

If the norms in the two spaces $\mathbb{V}$ and $\mathbb{W}$ are L^2-norms (as they are in the context of UQ), then the operator norm can practically be calculated as follows. By representing the elements of $\mathbb{V}$ and $\mathbb{W}$ as coordinate vectors with respect to orthonormal bases, the operator $\mathcal{A}$ can be represented as a matrix A. The operator norm of $\mathcal{A}$ is equal to the largest singular value of the matrix A.

3.2. Emulators

An emulator is a stochastic description of an uncertain function. Mathematically, an emulator is a random field over the parameter domain. In the following, we will use the terms *emulator* and *(random) field* synonymously. A one-dimensional random field is also called a stochastic process or random process.

A random field is a probability distribution over functions. So in analogy to random variables that can be thought of as a list of real numbers together with probabilities, a random field can be thought of as a collection of functions together with probabilities. To be useful as a surrogate model, a random field must have the property that the original model u is a realization of it.

Using random fields, we have to juggle multiple probability spaces: the one for the input parameters and the one for the random field. To avoid confusion, stochastic operators (E, Var, Cov) are equipped with indices to indicate over which probability space they are to be applied. For example, E_X denotes the expected value over input parameter X, while E_U denotes the expected value with respect to the field U.

Very often, the construction of a random field consists of the following three steps.

1. A parametric class of random fields is selected, from which u could be a realization. The parameters of such classes often include correlation length parameters and smoothness parameters and I refer to these parameters as *meta-parameters*.
2. One random field from this class is selected, in other words, the meta-parameters are fixed. This is done by using expert knowledge or by evaluating the model function on a small space-filling set of nodes $\mathcal{X} \subset \Omega$ and selecting the field parameters according to the model responses, e. g. via the maximum likelihood method [93]. The random field obtained in this step is called the *a priori* field which we denote by U_0. The a priori field encodes the knowledge about the structure of the model function, but not yet the actual function.
   Some authors recommend a validation of the a priori field [73]. Alternatively, instead of fixing the meta-parameters, one could model the meta-parameters as random variables themselves. The resulting random field will be a more complex mathematical object (compared to the field with fixed meta-parameters), but at the same time will be more flexible. In geostatistics, this approach is called Bayesian geostatistics [41, 76, 92, 106].
3. The a priori field is conditioned on model evaluations of the model function. These model evaluations include the initial space-filling set of nodes $\mathcal{X}$ as well as nodes selected later on. Conditioning on model evaluations means that only those field realizations are kept that interpolate the model responses. The result is an *a posteriori* random

field, which we denote by U_n, where n is the number of nodes.

The most frequently used emulators are Gaussian process emulators (gpe) [8, 20, 36, 91, 107, 109, 128, 159]. A random field U_0 is a gpe if, for any finite number of points $x_1, \dots, x_n \in \Omega$, the joint distribution of the random variables $U_0(x_1), \dots, U_0(x_n)$ is a multivariate normal distribution.

A gpe is fully described by its field mean function m_0 and the covariance function C_0:

$$\begin{aligned} m_0(x) &:= \mathrm{E}_{U_0}[U_0(x)] \\ C_0(x, x') &:= \mathrm{Cov}_{U_0}[U_0(x), U_0(x')], \end{aligned} \tag{3.18}$$

for any points $x, x' \in \Omega$. A common a priori choice is a second-order stationary field. Stationarity means that m_0 is constant and that C_0 is a function only of the difference between two points: $C_0(x, x') = C_0(x - x')$ [50]. The variance $\mathrm{Var}_{U_0}[U_0(x)] = C_0(x, x)$ is a representation of code uncertainty as a function of the input parameter x.

The covariance function expresses a smoothness assumption about the random field. For example, the realizations of a field with a Gaussian covariance function are infinitely differentiable, while an exponential covariance function leads to continuous, but non-differentiable realizations. A useful family of covariance functions is the Matérn family: it contains a smoothness parameter that determines how often the realizations are differentiable [42, 43, 63] and both the Gaussian and the exponential covariance function are special cases from the Matérn family.

If U_0 is a gpe, then the conditioned field U_n is again a gpe with updated mean and covariance function [50]. Let $x_1, \dots, x_n \in \Omega$ be a list of n nodes in the parameter domain and let $u(x_1), \dots, u(x_n)$ be the corresponding model responses. Define the residual vector

$$r = [u(x_1), \dots, u(x_n)]^\top - [m_0(x_1), \dots, m_0(x_n)]^\top,$$

the covariance matrix Q with

$$[Q]_{ij} = C_0(x_i, x_j)$$

and, for a point $x \in \Omega$, define the vector

$$q(x) := [C_0(x_1, x), \dots, C_0(x_n, x)].$$

The mean function and covariance function of the conditioned gpe U_n are given by:

$$\begin{aligned} m_n(x) &= m_0(x) + q(x)\,Q^{-1}r \\ C_n(x,x') &= C_0(x,x') - q(x)\,Q^{-1}q(x')^{\top} . \end{aligned} \tag{3.19}$$

In geostatistics, the procedure of conditioning a gpe is called Kriging and has a slightly different interpretation. While the conditioning step implicitly is the application of Bayes' theorem to a random field, Kriging interprets the same step as an estimation procedure. More precisely, the Kriging estimator is defined as the best linear unbiased estimator (BLUE). Both interpretations (conditioning and estimation) lead to the same result, presented in Eqs. (3.19). The Kriging variant presented here classifies as *simple Kriging* [24, 152]. Strictly speaking, simple Kriging requires the random field under consideration to have a zero mean. We obtain Eq. (3.19) if we apply the simple Kriging method to the residual function $u(x) - m_0(x)$ (because this residual function has a zero mean).

The conditioning of a random field also has a close connection to interpolation with radial basis functions: in some cases, both approaches result in the same surrogate model [100].

In connection with emulators, the set of nodes $x_1, \cdots, x_n$ is often called a *design of computer experiments* [128] because each evaluation of the model function is regarded as an experiment, in which the behavior of the model function is observed. In the context of emulators, I will use the term "design" and, in the context of response surface methods, I will use the term "sampling rule". Aside from the different contexts, both terms simply refer to the selection of nodes.

Similar to response surface models, we assume that the mean function and covariance function of an emulator are so fast to evaluate, that the only computational costs lie in the evaluation of the model function required to form the residual vector r. The overall computational costs are therefore determined by the number of nodes n.

If required, the mean function of an emulator can be used as a surrogate model. This approach is called *plug-in approach.* That way, however, one loses the variance and covariance information of the emulator. As I mentioned earlier, the variance of an emulator is an expression of code uncertainty, and as we will see later, it is beneficial to use this information.

3.3. Low-Fidelity Models

The computational costs of a model function can often be reduced if some of the model fidelity is sacrificed. This can be done by reducing the spatial or temporal resolution of the simulation or by neglecting some physical aspects in the model. Therefore, I will call low-fidelity models also *reduced models*. This term is not to be confused with reduced-order models [117, 129], which are not considered here. A reduced model can be thought of as a physically motivated surrogate model. Assuming that the model function is the correct, error-free representation of the processes under consideration, see Section 1.1, a reduced model introduces a *model error*.

Reduced models are faster than their high-fidelity counterpart (the original model function), but are usually not as fast as response surface models or emulators as presented in the previous two sections. Therefore, we cannot neglect the computational costs of a reduced model. Instead, we assume that a reduced model is faster than the model function by some fixed factor. Due to this speed up, we can afford larger sample sizes in any kind of sampling-based method and thereby reduce the stochastic error, see Section 2.1.4. In that perspective, the use of a reduced model is a trade between stochastic error and model error. Whether this trade is beneficial depends on the magnitude of both errors.

Part II.

Contributions

4. Optimal Sampling for Polynomial Response Surfaces

Most of the results presented in this chapter have been published in the International Journal for Uncertainty Quantifications under the title *An Optimal Sampling Rule for Nonintrusive Polynomial Chaos Expansions of Expensive Models* [132]. I am reusing parts of the text and figures from this publication by kind permission of the publisher Begell House.

In this chapter, I present the optimized stochastic collocation method (OSC). It is an optimal sampling rule for the calculation of PCE coefficients via stochastic collocation. As I stated in Section 3.1.2, the performance of a stochastic collocation method strongly depends on the chosen sampling rule.

My main goal behind OSC was to construct an *efficient* sampling rule in the sense that it should achieve an accurate response surface at a small computational effort. While efficiency is the final goal, it turned out to be useful to have a closer look at the following three aspects in the comparison of sampling rules:

Stability Polynomials are subject to the Runge phenomenon [126]: if nodes are distributed equidistantly in the domain, then the polynomial interpolant on these nodes tends to oscillate between the nodes. Therefore, nodes for a stable polynomial interpolation have to be more dense in the outer parts of the domain. The same effects holds for quadrature rules. Quadrature rules that are specifically adapted for the integration of polynomials (e. g. Gaussian Quadrature) typically have a higher node density in the outer parts of the domain. A good sampling rule should avoid oscillations in the response surface because otherwise an increasing computational effort does not guarantee an increasing accuracy.

Flexibility Since the model function is expensive, the sampling rule should be flexible in the number of nodes. This is similar to the choice of the ansatz space, see Section 3.1.2. A modeler can relatively freely select an ansatz space to control the number of required expansion coefficients. The modeler should have the same freedom in the number of nodes, as these are directly related to the computational costs. Higher-dimensional sampling rules often stick to certain structures (e. g. grids) and are therefore mostly not flexible in the number of nodes.

Versatility Since the input parameters might have statistical dependencies (see Section 2.1.1), the sampling rule should be able to handle these dependencies directly, without transforming the input parameters to statistically independent parameters.

Outside this chapter, the three terms stability, flexibility and versatility are usually not used with this specific meaning. I chose to given them these definitions so that the discussion in this chapter becomes more precise and easier to follow. But even more importantly, these three terms turned out as useful for formulating the shortcomings of existing sampling rules. As we will see in the numerical results later in this chapter, the reason why OSC is more efficient than other sampling rules is that it is stable, flexible and versatile, while all existing sampling rules lack at least one of these properties.

PCE coefficients can be calculated via the quadrature approach, the least-squares approach or possibly even other approaches, see Section 3.1.2. In all cases, I will call the points at which the model function is to be evaluated *nodes* and the method of choosing nodes (and weights) *sampling rule*.

4.1. Existing Sampling Rules and Their Shortcomings

Before I introduce my own sampling rule, let me present existing sampling rules and discuss them with respect to the three criteria stability, flexibility and versatility. I grouped the existing sampling rules into three groups:

grid-based rules, cubature rules and random or quasi-random rules.

Strictly speaking, a sampling rule is only complete if it is associated with a rule for calculating the expansion coefficients, see e. g. the quadrature approach and least-squares approach in Section 3.1.2. In the following descriptions I will shortly comment on the calculation rules that are commonly associated with the sampling rules.

Recall that the dimension of the parameter domain Ω is denoted by d.

4.1.1. Tensor Grids, Sparse Grids and PCM

Tensor grids and derived methods require that the set of admissible parameter values is a Cartesian product of one-dimensional sets, $\Omega = \Omega_1 \times \cdots \times \Omega_d$, and that the components of X are independently distributed. For each individual one-dimensional set $\Omega_i \subseteq \mathbb{R}$, nodes are determined according to a one-dimensional sampling rule. Then, all possible combinations of the nodes for the individual parameters are formed. If we select n_i nodes along each dimension $i \in \{1, \ldots, d\}$, then the tensor grid consists of $\prod_{i=1}^{d} n_i$ nodes. If the one-dimensional sampling rules are chosen to be stable (e. g. nodes from Gaussian Quadrature, Chebyshev nodes,...), then also the resulting tensor grid is stable. Tensor grids can be used both with the quadrature approach or the least-squares approach. If the one-dimensional quadrature rules are from Gaussian quadrature, then both approaches even lead to the same coefficients. The main downside of tensor grids is that they are not flexible and not versatile.

A sparse grid is a combination of multiple tensor grids, such that features of the model function in each coordinate direction can be captured well, while keeping the number of nodes lower than in the full tensor grid [7, 56]. The lower number of nodes is achieved by investing fewer nodes in cross-terms between the coordinates. For a fixed dimension and with increasing number of nodes, sparse grids have almost the same convergence behavior for integration as full tensor grids. However with increasing dimension, sparse grids of low order have increasingly large errors. This means that sparse grids in high dimensions are well suited only if the number of nodes is high as well. Sparse grids have been used as sampling methods for PCE [103, 168]. They are typically used with the quadrature approach. However, recent work shows that, rather than writing the PCE approximation as an integration problem and computing the integrals by a sparse grid quadrature, it is better to apply the sparse grid construction to the projection operator directly [32]. Similar to tensor grids, sparse grids are stable, but not flexible and not versatile.

The PCM is a heuristic, based on the full tensor grid [69, 70, 147, 156]. Because of its simplicity it is widely used [85, 87, 110, 139]. Aiming for an approximation of u with p terms [Eq. (3.8)], the PCM selects $n = p$ nodes from the full tensor grid, namely those with the highest weights in the associated full-grid quadrature rule, and performs a polynomial interpolation. In the selection of the nodes, it also has to be taken into account that the polynomial interpolation must be well-posed on these nodes. PCM is flexible, but generally not stable (we will see this in the numerical examples in this chapter). It is probably possible to use PCM in a versatile way (taking account of input dependencies), but I am not aware of any publication in which this has been done.

4.1.2. Monomial Cubature Rules

As a non-grid-based method, sampling based on monomial cubature rules has been proposed [140, 157]. The idea is to select nodes according to multidimensional quadrature rules with high polynomial degree. Wei et al. [157] present four different quadrature rules. These are of order 5 and 7 and are restricted to normally distributed input variables. These sampling rules are stable, but they are not flexible: the modeler can choose one of the four presented quadrature rules, but has no further control over the number of nodes. Furthermore, the sampling rules are not versatile as they are restricted to normally distributed variables.

The OSC sampling rule presented in this chapter follows the same general idea as monomial cubature rules. However, instead of presenting a fixed collection of quadrature rules, I developed a method for generating optimal quadrature rules for any order and for arbitrary distributions.

4.1.3. Random Sampling and Quasi Monte Carlo

As opposed to the deterministic methods stated above, it is also possible to sample randomly or quasi randomly. Random samples are commonly used with a least-squares approach.

Random sampling simply means that the nodes are drawn from the distribution of the input parameters, similarly to a Monte Carlo simulation [66].

Random sampling is flexible and versatile, but not stable unless the model function is oversampled (i. e. the number of nodes is chosen much larger than the number of coefficients to calculate) [66].

As a variance reduction method for random sampling, it is also possible to apply quasi Monte Carlo methods, such as Hammersley sampling [62, 66]. The Hammersley points generally have a lower discrepancy than randomly selected points. However, as we will see in the numerical examples, a lower discrepancy does not necessarily improve the stability of random sampling. Hammersley points are flexible, but not versatile being defined only for the uniform distribution on hyper-cubes of arbitrary dimension.

4.2. OSC - The Optimized Stochastic Collocation Method

In this section, I present the optimized stochastic collocation method (OSC). Based on the idea of monomial cubature rules, it is a method for generating quadratures for any polynomial order and for arbitrary distributions. The OSC method is formulated as an optimization problem with an objective function that is adapted to the efficient approximation of PCE coefficients.

4.2.1. Method Formulation

From Eq. (3.10) in Section 3.1.2, recall that a quadrature rule is of the form

$$\int_\Omega f(x)\,\mathrm{d}\mu(x) \approx \sum_{j=1}^{n} w_j f\left(x_j\right).$$

We define the exact integral operator $\mathcal{I}$ with respect to the measure μ:

$$\mathcal{I} : \mathbb{U} \to \mathbb{R} : f \mapsto \int_\Omega f(x)\,\mathrm{d}\mu(x). \tag{4.1}$$

Now we try to find the quadrature formula that is closest to $\mathcal{I}$ in some sense. For a given list of nodes $\boldsymbol{x} = (x_1, \ldots, x_n) \in \Omega^n$ and weights $w = (w_1, \ldots, w_n)$, we define the discrete quadrature operator

$$Q_{(\boldsymbol{x},w)} : \mathbb{U} \to \mathbb{R} : f \mapsto \sum_{j=1}^{n} w_j f\left(x_j\right). \tag{4.2}$$

A similar representation of integration and quadrature as operators has first been given by Xiu [166].

The key idea behind OSC is to find nodes and weights such that the quadrature operator $\mathcal{Q}_{(\boldsymbol{x}, w)}$ resembles the exact integration operator $\mathcal{I}$ as closely as possible. To do so, we measure the distance between the two operators with the operator norm of their difference.

The operator norm for $\mathcal{I} - \mathcal{Q}_{(\boldsymbol{x}, w)}$ is only defined if these operators are bounded. On the full space $\mathbb{U}$, boundedness is not guaranteed, so we restrict both $\mathcal{I}$ and $\mathcal{Q}_{(\boldsymbol{x}, w)}$ to a finite-dimensional test space $\mathbb{T}$, a subspace of $\mathbb{U}$. Recall from Section 3.1.3, that linear operators on finite-dimensional spaces are always bounded and we can then introduce the desired operator norm on $\mathfrak{L}(\mathbb{T}, \mathbb{R})$. In Section 4.2.3, I will explain how this operator norm can be evaluated in practice.

We are now ready to formulate the OSC method. The procedure for determining optimal nodes $\boldsymbol{x}^\star$ and weights $w^\star$ by OSC is:

1. Choose a finite-dimensional test space $\mathbb{T} \subseteq \mathbb{U}$ and the number of nodes n.
2. Determine the optimal nodes and weights according to

$$\left(\boldsymbol{x}^\star, w^\star\right) = \underset{\substack{\boldsymbol{x} \in \Omega^n \\ w \in [0, \infty]^n}}{\operatorname{argmin}} \left\| \mathcal{I} - \mathcal{Q}_{(\boldsymbol{x}, w)} \right\|_{\mathfrak{L}(\mathbb{T}, \mathbb{R})}^2 . \tag{4.3}$$

We can immediately make the following observations.

- This approach does not require us to discretize the optimization. While in the field of optimal spatial design, similar optimization problems are usually discretized and solved on a grid of candidate points, we regard the problem as a continuous optimization problem.
- The objective function is a multivariate polynomial. The functional form will be discussed in more detail in Section 4.2.3. The smoothness of the objective function suggests the use of gradient-based optimization algorithms.
- Minimizing the squared norm in step 2 is equivalent to minimizing the norm itself because the norm is nonnegative. By considering the square, the objective function becomes a sum of squares. This structure can be exploited by optimization algorithms, see Section 4.2.3.
- It may seem that, when the multi-dimensional integration problem is transferred into a multi-dimensional optimization problem, the level

of difficulty remains the same. However, the optimization can be done without evaluating the model function. Under the assumption that the model function is computationally expensive, the optimization can still be beneficial. This will be further discussed in Section 4.2.5.

- The optimization has $n\,(d+1)$ degrees of freedom. For a fixed polynomial degree and increasing dimension, the number of degrees of freedom grows faster than the number of polynomial terms. This is an important aspect in the discussion in Section 4.2.5.
- The ansatz space $\mathbb{P}$ is not entering the optimization procedure directly. However, the choice of $\mathbb{T}$ and n can only be done in a meaningful way if $\mathbb{P}$ is selected first. This is discussed in Section 4.2.2.
- The weights are forced to be non-negative. This has two reasons. First, the least-squares approach does only make sense if the weights are non-negative. Otherwise the squared residuals at some points in the parameter domain would be maximized rather than minimized. Second, numerical tests showed that if we allow negative weights, then the objective function has many local minima that are troublesome for the appropriate choice of optimization algorithms. The solutions in these local minima often have two or more nodes very close together with weights of large magnitude and opposite sign. By enforcing non-negative weights, the nodes are forced to spread in the domain.
- The nodes in the optimization are constrained to Ω^n. If Ω is just the support of the measure μ, then it might have an irregular shape. In this case, and if the numerical software behind u admits it, it is advantageous to select Ω larger, such that it is a Cartesian product of intervals $\Omega = \Omega_1 \times \cdots \times \Omega_d$. In the following, I call this an augmented support of μ. This augmentation has two advantages. First, the constraints are easier to implement in an optimization algorithm. Second, a bigger domain Ω potentially allows a smaller minimum in the optimization, and may help to increase the degree of the quadrature rule for the given the number of nodes[1].

[1]Here is a simple example to show that one can increase the degree of a quadrature rule by setting Ω to a larger set than the support of μ. Let μ be the uniform distribution on $[-2,-1]\cup[1,2]$. We seek to construct a quadrature rule that is exact for all linear polynomials $u\,(x) = ax + b$. If we select the domain to be just the support of μ, namely $\Omega = [-2,-1]\cup[1,2]$, then at least two nodes are needed. Additionally, an optimization on

I do not address the issues of existence and uniqueness of the optimum. In practice, we will often be sufficiently satisfied if we achieve a suitably low value of the error norm.

Please note that a similar approach to constructing quadrature rules has been taken by Charushnikov in the 1970s [22]. The difference of my work is that I suggest the use of numerical optimization methods, while older approaches tried to find the optimum in Eq. (4.3) analytically. To my knowledge, it is only possible to obtain optimal weights analytically, but not the nodes.

4.2.2. Choice of Test Space and Number of Nodes

As noted before, the choice of $\mathbb{T}$ and n has to be adapted to the structure of the ansatz space $\mathbb{P}$. I suggest the use of a polynomial test space $\mathbb{T}$. By approximating the model function by its PCE in the first place, we already assume that it can be well approximated by polynomials. Therefore, it makes sense to also construct the sampling rule to work well with polynomials.

Before we discuss how to select the dimension of $\mathbb{T}$, let me provide some basic considerations on relations between $\mathbb{T}$ and n. A trivial lower bound for the number of nodes is $n \geq p$, with $p = \dim \mathbb{P}$. Otherwise, the image of the discretized projection operator is only a lower-dimensional subspace of the ansatz space $\mathbb{P}$, and the n nodes are not even enough to distinguish the p different ansatz functions. This would lead to an effect called internal aliasing, which means that not even the elements of $\mathbb{P}$ can be projected correctly [30]. A trivial upper bound for the number of nodes is $n \leq t$, with $t := \dim \mathbb{T}$. With t nodes it is always possible to reduce the error norm in Eq. (4.3) to zero [34, 136].

In practice, we can select n much smaller than this upper bound. This is made plausible by the following consideration. In order to reduce the objective function Eq. (4.3) to zero, one has to satisfy t equations. The number of degrees of freedom in the OSC optimization is $n\,(d+1)$. If we choose $\mathbb{T}$ and n, such that

$$t = n\,(d+1)\,, \tag{4.4}$$

we can hope to just have enough nodes to be able to satisfy all t conditions. In the following, I call Eq. (4.4) the *degrees of freedom condition*

such a domain is tedious. If, however, we select $\Omega = [-2, 2]$ and if u can be evaluated everywhere on this interval, then we can find a quadrature rule with only one node, namely the expected value, which is 0 here.

(DOF-condition). This condition, however, has to be understood as a rule of thumb. Both cases exist where n has to be chosen greater or can be chosen smaller[2].

I suggest one of the two following approaches for choosing the test space and n. I call these approaches the *rigorous approach* and the *minimal approach*.

Rigorous Approach Let us assume that the model function itself lies in the ansatz space, $u \in \mathbb{P}$. Then, it is reasonable to require that the surrogate model is exact, i. e. $\tilde{u} = u$. Inserting this requirement into either the quadrature approach [Eq. (3.11)] or the least-squares approach [Eq. (3.12)], we obtain integrals over products of two ansatz functions. Roughly speaking, this means that a quadrature rule must be exact up to twice the degree of u in order to guarantee that internal aliasing does not occur. As a result, we define the test space to be the span of all products of two ansatz functions:

$$\mathbb{T} = \operatorname{span}\left\{\Psi_i \Psi_j \mid 1 \leq i, j \leq p\right\} . \tag{4.5}$$

We then select n large enough so that the operator norm is reduced to zero. A practical procedure is to first select n according to the DOF-condition [Eq. (4.4)] and numerically perform the optimization. If the smallest found value of the objective function is not small enough, then n can gradually be increased until it is large enough to reduce the objective function to a sufficiently low value.

Minimal Approach We set the number of nodes to the trivial lower bound: $n = p$. Then we select an appropriate test space. Now the quadrature rule is, in general, not able to calculate all necessary integrals exactly.

[2]The following two examples show that the DOF-condition can both under- and overestimate the number of nodes required for exact integration.

The first example is described by Radon [118]. For a uniform distribution with $d = 2$ and $\mathbb{T} = \operatorname{span}\left\{1, x_1, x_2, x_1^2, x_1x_2, x_2^2\right\}$, it is $t = 6$ and the DOF-condition suggests choosing $n = 2$. However, no matter where the two nodes are placed in the domain, there always exists one strictly non-negative function in $\mathbb{T}$ that is zero in both nodes. Thus, no quadrature rule with $n = 2$ exists that is exact for all functions in $\mathbb{T}$.

For the second example, assume a two-dimensional tensor product of Gaussian quadrature rules with two nodes in each direction on a separable measure, $d = 2$, $n = 4$ and the DOF-condition suggests that such a quadrature rule is exact for a test space of dimension $n\,(d + 1) = 12$. In fact this quadrature rule is exact for all polynomials up to order 3 in each coordinate direction, which yields $t = 16$, a higher order of accuracy than the DOF-condition suggests. Tensor products of Gaussian quadrature rules have the property to be exact for $t = n \cdot 2^d$ test functions because the factor of 2 multiplies for each dimension.

The test space can be chosen either as in Eq. (4.5) or according to the DOF-condition [Eq. (4.4)]:

$$\mathbb{T} = \operatorname{span}\left\{\Psi_1, \ldots, \Psi_{n(d+1)}\right\}. \tag{4.6}$$

The former treats all coordinate directions of Ω according to their importance in $\mathbb{P}$. The latter simply cuts off the ONB after $n\,(d+1)$ terms, which means that some coordinate directions are slightly preferred over others. The word *minimal* in *minimal approach* refers to the minimality of n, not of $\mathbb{T}$.

4.2.3. Implementation Details

If the test space is chosen to be a polynomial space, a relatively simple expression for the squared operator norm $\left\|\mathcal{I} - \mathcal{Q}_{(\boldsymbol{x},w)}\right\|^2_{\mathfrak{L}(\mathbb{T},\mathbb{R})}$ can be derived. As before, we define $t = \dim \mathbb{T}$ and represent the elements of $\mathbb{T}$ as coordinate vectors with respect to the ONB $\Psi_1, \ldots, \Psi_t$. Trivially, $\mathbb{R}$ is a space of dimension 1 and its elements are represented as themselves. Formally, the corresponding basis is 1, which also constitutes an ONB. Accordingly, both operators $\mathcal{I}$ and $\mathcal{Q}_{(\boldsymbol{x},w)}$ can be represented as $1 \times t$ matrices. We denote the matrix representation of an operator $\mathcal{A}$ with respect to basis Ψ as ${}_{\Psi}\mathcal{A}$. Since we represent everything with respect to ONBs, the operator norm in $\mathfrak{L}\,(\mathbb{T}, \mathbb{R})$ can be computed as the 2-norm of its matrix representation, which is degenerated to a vector in this case.

For the matrix representation of $\mathcal{I}$ with respect to Ψ, we make use of the relationship in Eq. (3.4) to find

$$\begin{aligned} \mathcal{I}\Psi_1 &= 1 \\ \mathcal{I}\Psi_i &= 0, \quad \text{for } i > 1. \end{aligned} \tag{4.7}$$

Therefore,

$${}_{\Psi}\mathcal{I} = \mathrm{e}_1 = (1, 0, \ldots, 0)\,. \tag{4.8}$$

By the definition of $\mathcal{Q}_{(\boldsymbol{x},w)}$ [Eq. (4.2)], we find

$$\mathcal{Q}_{(\boldsymbol{x},w)}\Psi_i = \sum_{j=1}^{n} w_j \Psi_i\left(x_j\right) \tag{4.9}$$

and then

$${}_{\Psi}\mathcal{Q}_{(\boldsymbol{x},w)} = \left(\boldsymbol{\Psi}\left(\boldsymbol{x}\right) w\right)^{\top}. \tag{4.10}$$

with the $t \times n$ matrix

$$\boldsymbol{\Psi}(\boldsymbol{x}) := \begin{bmatrix} \Psi_1(x_1) & \dots & \Psi_1(x_n) \\ \vdots & & \vdots \\ \Psi_t(x_1) & \dots & \Psi_t(x_n) \end{bmatrix}$$

We can calculate the squared operator norm of $\mathcal{I} - Q_{(\boldsymbol{x}, w)}$ with

$$\left\| \mathcal{I} - Q_{(\boldsymbol{x}, w)} \right\|^2_{\mathfrak{L}(\mathbb{T}, \mathbb{R})} = \left\| \mathrm{e}_1 - (\boldsymbol{\Psi}(\boldsymbol{x})\, w)^\top \right\|^2_2, \tag{4.11}$$

which is a sum of squares. Together with the fact that the objective function is a multivariate polynomial and therefore smooth, this structure can efficiently be exploited with Gauss-Newton type optimization algorithms. An analytical implementation of the partial derivatives of the objective function can be used to further accelerate the optimization.

4.2.4. Properties of the Method

In this section we address two properties of the OSC method. First, with OSC it is straightforward to recycle nodes to form nested sampling rules. Second, OSC is a generalization of some existing and well-known quadrature rules.

Recycling of Nodes

With OSC it is possible to recycle nodes in the sense of nested integration rules. Assume we have already performed some model evaluations in earlier work. A fixed list of nodes $x_1, \dots, x_n$ and the model responses $u(x_1), \dots, u(x_n)$ are given. Next we want to add m more nodes $x_{n+1}, \dots,$ x_{n+m}, such that the quadrature rule with all $n + m$ nodes is an optimal extension of the given n nodes.

The reuse of nodes by OSC is straightforward. We use exactly the same objective function as before [Eq. (4.3)], only we fix the first n nodes in the optimization. The number of degrees of freedom is now $n + m(d + 1)$: each of the new nodes has $d + 1$ degrees of freedom, while the recycled points are free only in their weights. By recycling nodes we can increase the order of the quadrature rule, but the contribution of recycled nodes (in terms of degrees of freedom) is reduced from $n(d + 1)$ to n. Therefore, recycling of nodes becomes less effective in higher dimensions. The test space $\mathbb{T}$ can be selected analogously to the suggestions in Section 4.2.2. In this case, the DOF-condition has a slightly different form:

$$t = n + m(d + 1) \tag{4.12}$$

Two possible applications for the reuse of nodes are:

1. The test space $\mathbb{T}$ changes. For example, a lower-order PCE can be used as part of an error estimator, to predict whether a higher-order PCE is necessary. In this approach, polynomial response surfaces of increasing degree are constructed until an error estimator indicates that the degree is high enough. For these methods, it is highly desirable to reuse nodes. This idea is analogous to nestedness in quadrature rules. By reusing nodes, we construct a set of nested quadrature rules.
   A possible application is in sparse PCEs. For example, Blatman and Sudret propose an adaptive sparse polynomial chaos approximation, using a sequential experimental design [15]. Such sequential design could be improved by incorporating information about previous nodes in each iteration.
2. The measure μ of the parameter distribution changes. This is the case, for example, when Bayes' theorem is applied for parameter inference, see Section 2.2. After incorporating measurement data into the prior knowledge of the distribution, one obtains a posterior distribution that differs from the prior. The old nodes are generally not placed optimally with respect to the new measure and it is desirable to add more nodes if the modes of prior and posterior lie far apart [111]. Other situations where the applied measure changes are, for example, the so-called shifted PCE and the windowed PCE [113], or simply a change of mind in the definition of the prior.

OSC as a Generalization of Known Quadrature Rules

A couple of known quadrature rules are special cases of OSC: they minimize the objective function in Eq. (4.3) for certain choices of the test space. I present a selection of three such known types of quadratures rules. In numerical tests, which are not further reported here, I was able to confirm that these quadrature rules are not only theoretical minima of the objective function, but can also practically be found by numerical optimization.

Gaussian Quadrature (GQ) GQ rules are one-dimensional quadrature rules with the highest possible integration order [39, 60]. With n

nodes, they integrate the first $2n$ monomials $1, x, x^2, \ldots, x^{2n-1}$ exactly. This number of monomials satisfies the DOF-condition. To reproduce the nodes and weights of a GQ rule via OSC, we select a number n and then choose the test space according to the DOF-condition. By definition, the nodes of GQ minimize the objective function in Eq. (4.3). We do not have to restrict ourselves to a specific GQ rule, e. g. Gauss-Legendre, Gauss-Hermite, etc. Instead, the measure μ can be chosen freely, as long as the ONB of the test space can be constructed. OSC is also capable of reproducing tensor products of Gaussian quadrature if the underlying measure is separable and the ansatz space is chosen to be a tensor product polynomial space. For non-separable measures (i. e., for statistically dependent input parameters), OSC deviates from GQ rules, and provides more problem-adapted non-tensor clouds of nodes, see Section 4.3.3. In this sense, OSC is a generalization of GQ.

Kronrod Extensions and Gauss-Kronrod Rules Given a one-dimensional quadrature rule with n nodes, its Kronrod extension is the nested quadrature rule with additional $n + 1$ nodes that has the highest possible degree [115]. A Gauss-Kronrod quadrature rule is a Kronrod extension of a GQ rule. When constructing a Kronrod extension, there are $3n + 2$ degrees of freedom. Each old node yields one degree of freedom, namely its weight, while the $n + 1$ new nodes are free in location and weights and thus have two degrees of freedom each. To construct a Kronrod extension with OSC, we make use of the ability to recycle nodes. Starting from an n-node integration rule, we select $\mathbb{T} = \{\Psi_1, \ldots, \Psi_{3n+2}\}$, which is according to the modified DOF-condition [Eq. (4.12)] with $m = n + 1$. If the Kronrod extension exists, then, by definition, it globally minimizes our objective function [Eq. (4.3)]. Newtons method has already been used to numerically find Gauss-Kronrod rules [21] and general Kronrod extensions [54]. OSC coincides with the approach in these two papers if a Gauss-Newton algorithm is used for the minimization of Eq. (4.3). The main difference is that OSC is formulated for arbitrary dimensions and arbitrary numbers of additional nodes, while Kronrod extensions are one-dimensional by definition and always add $n + 1$ nodes to an n-node quadrature rule. Thus, it is justified to call OSC a generalization of Kronrod extensions.

General Multi-Dimensional Monomial Quadrature Rules Last, any general multi-dimensional monomial quadrature rule is a special case of the OSC method. A monomial quadrature rule is a quadrature rule that exactly integrates polynomials up to a certain degree. Research on such rules goes back to Radon in 1948 [118] and Stroud in 1971 [136]. By construction, a monomial quadrature rule attains an operator norm of 0 in Eq. (4.3) if the test space is chosen correctly. This means that if the numerical optimization is successful, then OSC can either reproduce these quadrature rules from literature, or we would find a quadrature rule that is different, but achieves the same polynomial degree with the same number of nodes. Numerical tests within my study revealed that in some cases there exists a continuum of quadrature rules that minimize the operator norm to 0, e. g. when the stochastic domain Ω and the measure μ are rotationally symmetric.

4.2.5. Limitations

The OSC has an important limitation: to obtain the nodes and weights, we need to solve a high-dimensional optimization problem. The dimensionality of the optimization problem is $n\,(d+1)$, and so increases with both the dimension of the parameter domain and the number of nodes. The applicability of OSC is restricted by the availability of efficient and robust optimization algorithms.

It is worth pointing out that, for a fixed polynomial degree and increasing dimension, the dimensionality of the optimization grows faster than the number of needed nodes. That means, no matter how expensive the model function is, with increasing dimension, there will be a point at which the optimization becomes more time consuming than additional model evaluations.

In such case, it might be more efficient to use a simpler integration rule and accept a non-minimal number of nodes. Most other rules, like sparse grid rules, are much simpler in their construction than OSC and the set of nodes can be determined explicitly and easily.

Another issue is the practical problem of finding the global minimum. The objective function is a multivariate polynomial and it can be expected to have local minima. The problem of local minima can be tackled by using multi-start optimization algorithms or global search algorithms. However,

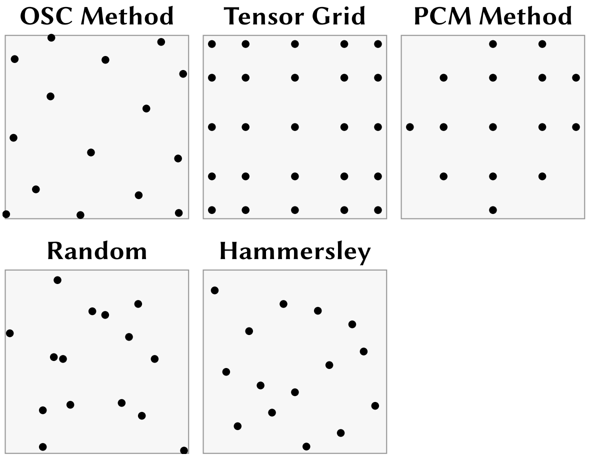

Figure 4.1.: Nodes generated by different sampling rules for the uniform distribution on $[-1, 1]^2$

even if the optimization is repeated a couple of times, there is no guarantee that the global optimum has been found, unless the objective function has been reduced to its lowest attainable value, i. e., zero. In Section 4.3.5, I report an experiment about the practical computing time of the optimization and the necessary number of multistarts.

4.3. Numerical Experiments

The following numerical experiments are meant to demonstrate that the OSC method is stable, flexible and versatile. In all experiments, polynomial coefficients were calculated using the least-squares approach, see Eq. (3.12). The results from the quadrature approach [Eq. (3.11)] are not shown in this chapter. In almost all experiments, the least-squares approach was equally accurate or better than the quadrature approach, regardless of the used sampling rule.

4.3.1. Flexibility

The first couple of experiments demonstrate the flexibility of OSC. Recall that the modeler is relatively free in choosing a specific polynomial basis for the expansion of the model function. By my definition, a sampling method is called flexible if it offers the same freedom in selecting the number of nodes.

In this experiment we consider a uniform distribution on the domain $\Omega = [-1, 1]^2$. I discretized this distribution with a discrete uniform distribution on a regular 100×100 grid. Please note that this discretization does not introduce any discretization errors because *all* of the computations involved (i. e., orthonormalization of the polynomial basis, calculation of the reference solution and of the errors) were performed on this discrete distribution. So instead of solving the original problem (on the continuous distribution) with errors, we solve a slightly different problem (on a discrete distribution) exactly.

We compare the following five sampling rules.

1. OSC,
2. tensor grids constructed from the Gauss-Legendre quadrature,
3. PCM,
4. random sampling and
5. Hammersley sampling.

To give a visual impression, Figure 4.1 shows nodes according to these five sampling rules for the construction of a two-dimensional polynomial of total degree 4. The tensor grid for this task has 25 nodes, while all other sampling methods require only 15 nodes.

Constant Polynomial Degree

We consider the model function $u(x_1, x_2) = \exp(x_1 + x_2)$ and approximate the model function by a polynomial of total degree 4, so the expansion has $p = 15$ terms. With each sampling method, we construct sampling rules of different sizes and calculate the approximation error for the 15 coefficients. The approximation errors are shown in the upper plot in Figure 4.2.

First of all, the three sampling methods PCM, random sampling and Hammersley sampling converge at a slower rate than OSC and the tensor grid. The OSC method and the tensor grid show a faster convergence behavior.

The only difference is that OSC starts with a smaller minimal sample size than the tensor grid: here, 15 nodes instead of 25.

If we compare the total error (=approximation error plus truncation error) of the OSC method and the tensor grid, see Figure 4.2, lower plot, we can see that the small approximation error of the tensor grid becomes invisible compared to the relatively large truncation error of the degree 4 polynomial. Choosing a sample with more than 30 nodes does not significantly improve the accuracy. Since the tensor grid is not flexible enough to provide a sample smaller than the 25 nodes, it follows that the two best samples for this test case are the two smallest OSC samples with 15 and 21 nodes, respectively.

Increasing Polynomial Degree

In practice, it does not make much sense to increase the number of nodes without increasing the number of expansion terms as well. Otherwise, the truncation error dominates the total error as seen in the lower plot in Figure 4.2. For a reasonable result, the polynomial degree has to be increased together with the number of nodes. Therefore, we repeat the previous experiment but increase both the number of expansion terms p and the number of nodes n. The total polynomial degree in this experiment varies between 1 and 8. In two dimensions, a total degree polynomial of degree g has $p = (g + 1)(g + 2)/2$ terms. For PCM, random sampling and Hammersley sampling, the minimal number of nodes $n = p$ is chosen. The OSC is constructed with the rigorous approach (see Section 4.2.2), which results in slightly more nodes than the minimum. The minimal tensor grid has $(g + 1)^2$ nodes, which is approximately twice as many as there are terms in the expansion.

The approximation error and the total error observed in this experiment are shown in Figure 4.3. Now, the approximation error of the individual data points in the plot are not directly comparable, as each expansion has a different number of terms. If we compare data points that belong to the same degree of expansion (e. g. the last data point of each plot) then we see that tensor grid rules are much more accurate than other methods (an error of 10^{-9} versus 10^{-6}) but also need more nodes. The behavior of the total error (Figure 4.3, lower plot) shows that the OSC overall is more useful. This is because the high approximation accuracy of the tensor grid rule is compromised by the relatively large truncation error: while each tensor grid is

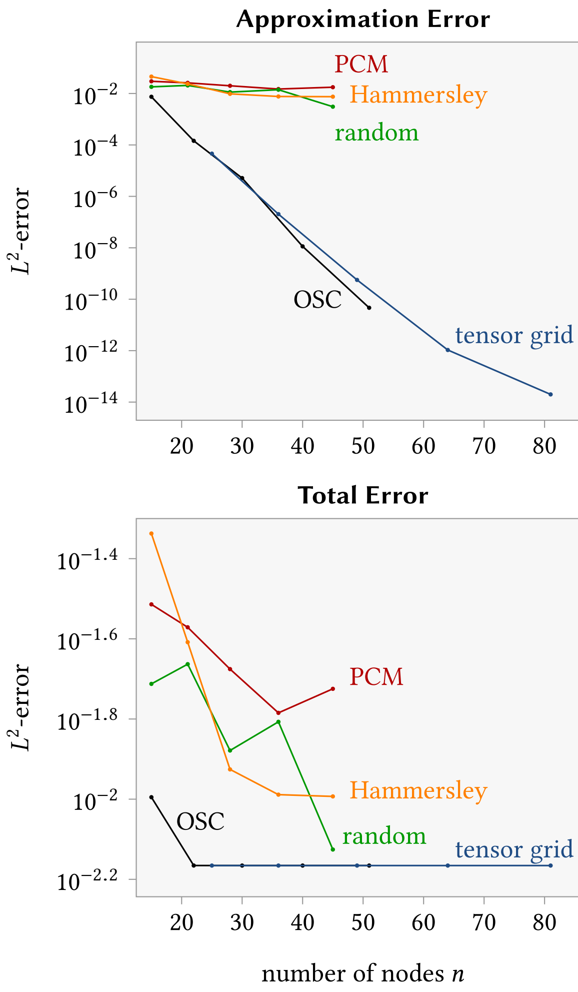

Figure 4.2.: Approximation error and total error for the approximation of the function $u(x_1, x_2) = \exp(x_1 + x_2)$ by a polynomial of constant degree 4 over a uniform distribution

very accurate by itself, the total error is always dominated by the truncation error. Therefore, the additional nodes in the tensor grid are spent on an additional approximation accuracy that does not significantly improve the overall accuracy.

These experiments show that the only reason why tensor grid rules perform worse than the other rules, is that they are not flexible. OSC and the other three methods are flexible.

4.3.2. Stability

In this experiment, we test the different sampling rules for their stability. We repeat the previous experiment, but with a different model function $u(x_1, x_2) = 1/\left(1 + 5x_1^2 + 5x_2^2\right)$. This function can be thought of as a two-dimensional version of the Runge function. The Runge function is a typical example for a function that, on an equidistant set of nodes, cannot be interpolated by polynomials in a stable way [126]. The total errors are shown in Figure 4.4. While OSC and the tensor grid method improve their accuracy with an increasing sample size, the three other sampling rules (PCM, random sampling and Hammersley sampling) have an increasing error. This is in line with the fact that the Runge function is difficult to interpolate. The nodes are not well-spread in the domain to reliably approximate the growing number of coefficients. These three sampling rules are flexible, but not stable if used with the minimal number of nodes. The OSC method and tensor grids are both stable. This experiment shows that stability is an absolute requirement for a sampling rule: if a sampling rule is not stable, it should be avoided altogether.

4.3.3. Versatility

We now compare the different sampling rules in their ability to handle dependent input parameters. We select the following distribution:

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} R \cdot \cos(\theta) - 1 \\ R \cdot \sin(\theta) - 1 \end{bmatrix} \tag{4.13}$$

with two random variables $R \sim \mathcal{U}([1, 2])$ and $\theta \sim \mathcal{U}([0, \pi/2])$. The values of X are again in the interval $[-1, 1]^2$, and the support of the distribution is a quarter of a ring around the point $(-1, -1)$ with radii 1 and 2. Similarly

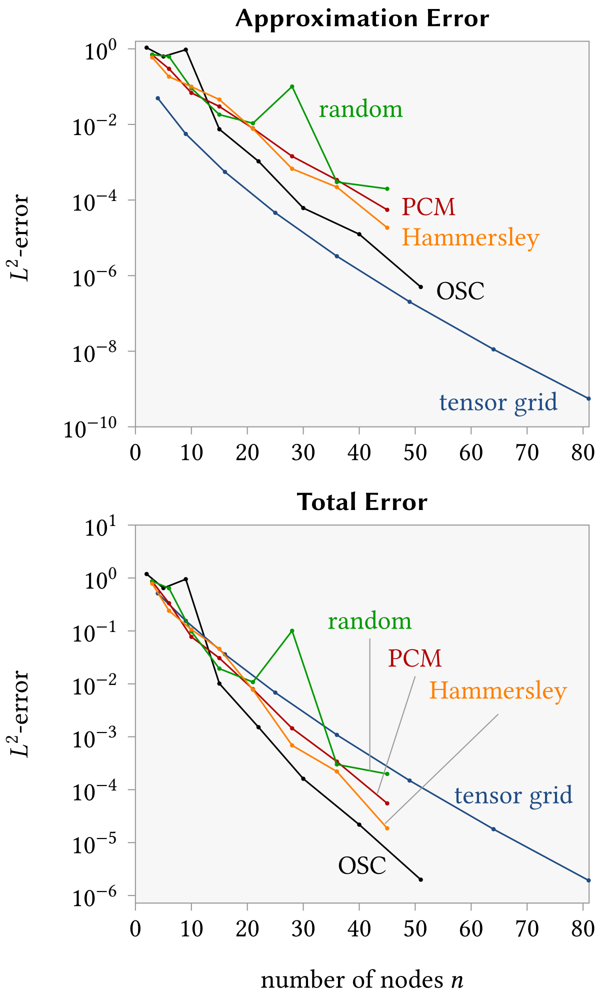

Figure 4.3.: Approximation error and total error for the approximation of the function $u(x_1, x_2) = \exp(x_1 + x_2)$ by polynomials of degrees between 0 and 8 over a uniform distribution

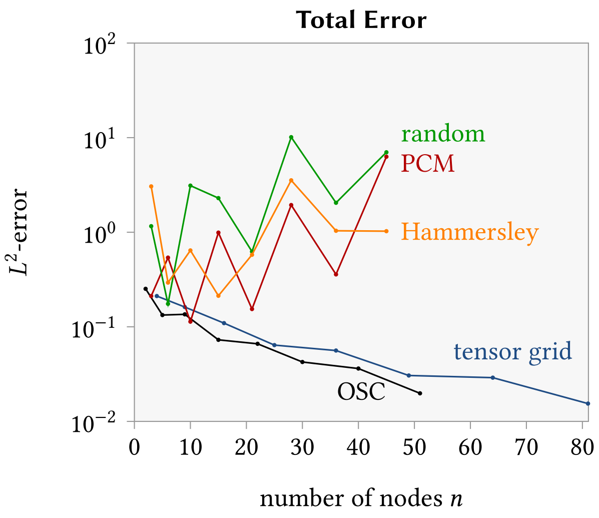

Figure 4.4.: Total error for the approximation of the Runge-type function $u(x_1, x_2) = 1/\left(1 + 5x_1^2 + 5x_2^2\right)$ by polynomials of degrees between 0 and 8 over a uniform distribution

to the previous experiments, the distribution was discretized by a sample of size 10 000, and all calculations were done one this discrete distribution. Therefore, we do not get additional discretization errors.

For a visual impression, Figure 4.5 shows the nodes determined by the different sampling rules for a polynomial of total degree 5. The tensor grid was constructed by ignoring the joint distribution altogether. One-dimensional quadrature rules we constructed from the two marginal distributions and then all possible combinations were used. The PCM nodes were constructed similarly. OSC can take full account of the dependency. Therefore, the nodes of OSC all lie within the support of the distribution of X, expect for one outlier. This is noteworthy because, in the optimization, no explicit constraints were imposed, following the idea of an augmented support for the integration (see last remark in Section 4.2.1). The outlier can be explained by the fact that the functions of the test space $\mathbb{T}$ are defined outside the support as well. Since the functions are smooth, even nodes outside the support are informative about the model function u.

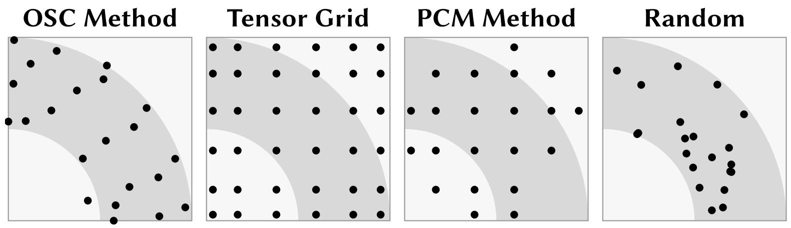

Figure 4.5.: Nodes generated by different sampling rules for a dependent distribution. The gray area indicates the support of the distribution.

As before, we define the model function $u(x_1, x_2) = \exp(x_1 + x_2)$, and approximate it with polynomials up to a total order of 6. The total errors are shown in Figure 4.6. Again, OSC is the most efficient sampling rule.

There is one important difference between this convergence plot and the previous ones. Previously, OSC was about equally accurate as the tensor grids, but more efficient, due to the smaller sample sizes. In this experiment, OSC is both more accurate and faster than the tensor grids. This demonstrates that OSC is more versatile than tensor grids.

In this example, the dependency in the parameter distribution could be removed by transforming the model function to the (R, θ)-space. This is not done here for the reasons discussed in Section 2.1.1.

4.3.4. Nested Sampling Rules

To demonstrate how OSC is able to produce nested node sets, we mimic a case of Bayesian updating by using two different distributions. In the first step, we generate a set of nodes for a prior distribution, which we select to be $\mathcal{N}(0, 1)$ for both parameters independently. In the second step, we add more nodes using a fictitious posterior distribution $\mathcal{N}\left(1, 0.5^2\right)$. These distributions are selected by hand, but they could occur when Bayes' theorem is applied to a linear model with normally distributed measurement errors. In Figure 4.7, the two distributions are indicated by the gray shading. The distributions are chosen such that some of the prior nodes lie in the high-probability region of the posterior, which means that some interactions between the prior and the posterior node cloud can be expected.

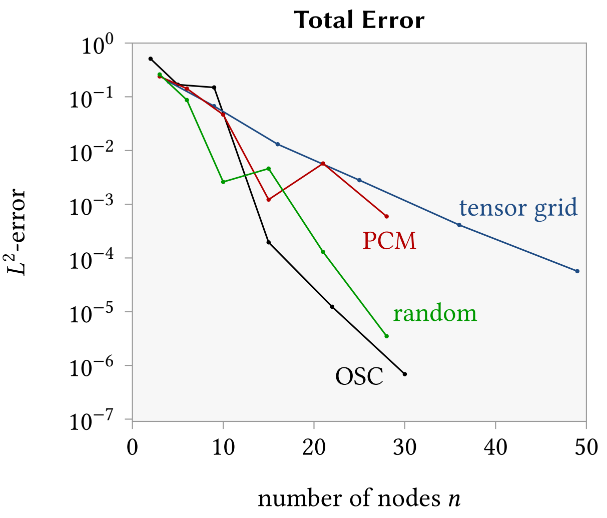

Figure 4.6.: Total error for the approximation of the function $u(x_1, x_2) = \exp(x_1 + x_2)$ by polynomials of degrees between 0 and 6 over a distribution with dependencies

We generate twelve nodes for the prior with a test space of polynomials up to degree 7. For the posterior, we add eleven nodes with a test space of polynomials up to degree 8. The nodes are shown in Figure 4.7. On the left, the twelve prior nodes are shown. On the right, the prior nodes are shown in white and the newly added nodes are shown in black.

As expected, the newly added nodes interact with the prior nodes and leave gaps around them. It is clear that the 23-node rule is not the best integration rule one can obtain with 23 nodes, but it is the best possible extension of the preset twelve nodes. If the posterior distribution was known beforehand, then we could have generated a 15 node rule with the same test space. By recycling the old nodes, we only had to add 11 new nodes, saving 4 nodes.

4.3.5. Optimization Effort and Robustness

The dimension of the optimization problem is $n(d+1)$, and increases with both dimension d and number of nodes n. To provide a rough idea about

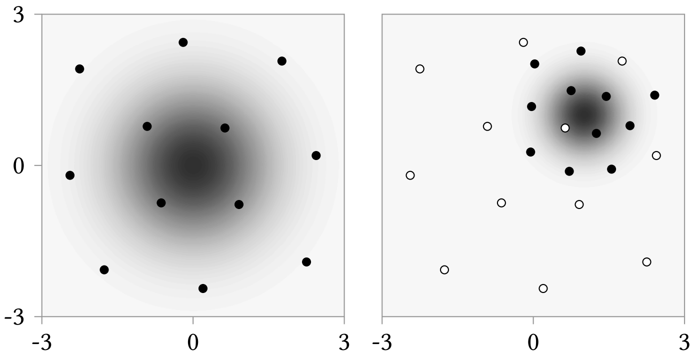

Figure 4.7.: Nested sampling rules: prior (left) and posterior (right) distribution and accordingly placed nodes

Dimension	Degree	Number of nodes	Degrees of freedom	Optimization time [s]
1	1	2	4	0.02
	5	6	12	0.07
	10	11	22	0.21
2	1	3	9	0.03
	5	21	63	0.26
	10	66	198	2.14
5	1	6	36	0.29
	5	252	1 512	4 020.80
10	1	11	121	51.17
	2	66	726	2 171.10

Table 4.1.: Optimization time for OSC for different polynomial degrees in different numbers of dimension.

the computing time, we perform the optimization in different dimensions (number of random input variables) and for various numbers of nodes

For all of the following calculations, we assume a uniformly distributed input X in a d-dimensional unit cube $[-1, 1]^d$. For various dimensions, we construct ansatz spaces of different total degrees. An ansatz space of polynomials up to total degree g in d dimensions has a basis of

$$p = \frac{(d + g)!}{d!g!} \tag{4.14}$$

ansatz functions [81]. For each case, we perform optimizations with the minimal number of nodes $n = p$ and the test space according to Eq. (4.5) (see Section 4.2.2). To get a robust estimate, the computing time is averaged over 10 optimization runs for each case. The initial sample for each optimization is drawn randomly from the distribution of X. Our implementation uses the objective function as derived in Section 4.2.3. Additionally, the partial derivatives of the objective function are implemented analytically. For the optimization, Matlab's function `lsqnonlin` was used. It is an implementation of an interior trust region algorithm which uses a second-order Taylor expansion of the objective function [29]. The optimization was performed on one core of a desktop computer with 3.10 GHz. Results are summarized in Table 4.1 and Figure 4.8 shows a plot of the optimization time in terms of degrees of freedom. Data points from different dimensions are put together in this plot. In the range of up to 100 degrees of freedom, the optimization time increases approximately linear (with a slope of approximately 1 in the log-log-plot). Beyond that point, the slope increases, indicating roughly a quadratic or faster increase. However, our data are sparse in this area.

In the explored range, the optimization time is in the order of seconds, and only one of the cases takes more than one hour. Recall that, in the introduction, I made the assumption that the model function is so expensive that the computational costs of preprocessing steps can be neglected. For the cases considered here, with a relatively low dimension of the parameter domain and a moderate polynomial degree, this assumption seems to be justified. If, however, the parameter domain is higher-dimensional and if the polynomial degree is further increased, the optimization will take more time and the assumption should be reconsidered.

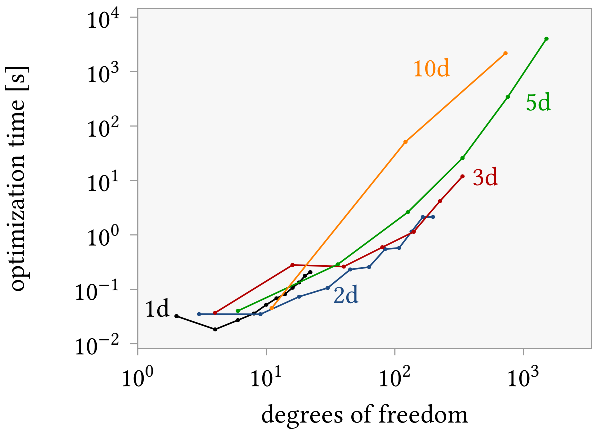

Figure 4.8.: Optimization time in terms of degrees of freedom.

Robustness

Finally, we investigate the numerical robustness of the optimization. How sensitive is the optimization result to the random initial conditions? We compare three cases, all of dimension 2 and all with a test space of polynomials up to degree 8. In the first case, the underlying distribution is uniform on the unit square. In the second case, we select a bimodal Gaussian mixture distribution: with a probability of 50/50 the random parameter is sampled from one of two normal distributions $\mathcal{N}((0,0), I)$ and $\mathcal{N}((3,3), I)$. In the last case, the parameters are again uniformly distributed, but this time we generate 6 random points and insert them into the optimization as fixed nodes for node recycling (see Section 4.2.4).

For each of the three cases, we repeat the optimization 1000 times and record the optimization time and the achieved value of the objective function. Figure 4.9 shows scatter plots of these data. In the left plot, we can see that the objective function has three distinct local minima. The number of runs that reached the global minimum is 420 out of 1000. In the second case, the objective function seems to have many more local minima and the number of successful runs is 278 out of 1000. In the third case, very many runs did not find the global minimum and only 55 out of the 1000 runs reached the optimal point.

These results indicate that, for multimodal distributions and with node recycling, the optimization becomes practically more difficult. Local search algorithms as the one used here do not guarantee to find the global minimum. Consequently, the computational effort for finding optimal nodes is the product of the average optimization time and the number of necessary multistarts. Furthermore, in practice we have to accept that we can not always find the optimal quadrature rule. For a practical approach it might be sufficient to use a different sampling rule as the initial guess for the optimization. The resulting sample might not be the optimal sample, but it is guaranteed to be better than the initial guess, or at least equally good.

In all three cases reported in Figure 4.9, the computing time varied mainly within one order of magnitude with a few outliers in case 2 and 3, which are slower by a factor of about 100 compared to the fastest run.

4.4. Beyond Quadrature and Least Squares

As an outlook for future research, I will formulate a possible generalization of the OSC method. Unlike the rest of the chapter, this content has not been published in the International Journal for Uncertainty Quantification.

In the current formulation, the OSC method has a conceptual downside: the method finds nodes and weights, but does not provide a clear rule for how to calculate the PCE coefficients from these. The modeler still has to decide between the quadrature approach and the least-squares approach and, as I mentioned in Section 3.1.2, I am not aware of any scientific work that provides a practical aid in this decision by directly comparing these two approaches.

This downside can be avoided by the following procedure. The OSC method is based on optimization anyway, so we can use the optimization procedure itself to determine the optimal calculation rule for the PCE coefficients. In Eq. (3.13) in Section 3.1.2, I provided a general formulation of all possible linear calculation rules: $\mathcal{P} = \Psi A \mathcal{E}_{\boldsymbol{x}}$. If we take this formulation as a starting point, we only need to find a suitable objective function and then the resulting optimization problem will able to find both the optimal nodes $\boldsymbol{x}$ and the optimal calculation rule A. Such an optimal calculation rule might either coincide with one of the known approaches (quadrature or least-squares) or might be entirely new. In the following, I will provide some mathematical details and propose two possible objective functions.

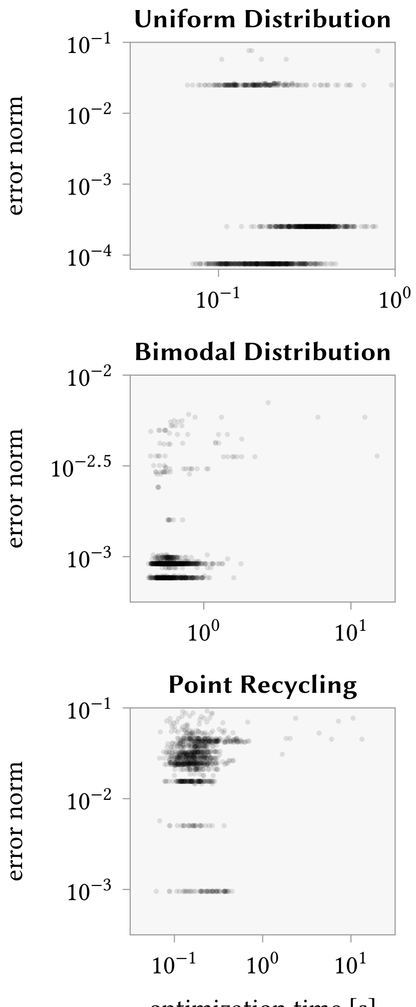

Figure 4.9.: Scatter plot of achieved value of the objective function versus optimization time

From Section 3.1.2 recall that a general form of a response surface operator with a fixed ansatz space is $\Psi A \mathcal{E}_{\boldsymbol{x}}$, see Eq. (3.13). The matrix A encodes the rule for calculating the PCE coefficients from the model responses. The quadrature approach and the least-squares approach are special cases of this operator in the sense that the matrix A can be chosen such that the whole operator coincides with either of the two approaches. Starting with this general form of an operator, we can optimize both the nodes $\boldsymbol{x}$ and the calculation rule A so that the operator $\Psi A \mathcal{E}_{\boldsymbol{x}}$ becomes as similar to the best-approximation operator as possible.

We define the operator that maps a function to its best approximation in $\mathbb{P}$ as

$$\mathcal{P}^\star : \mathbb{U} \to \mathbb{P} : u \mapsto \sum_{i=1}^{p} \langle u, \Psi_i \rangle \, \Psi_i .$$

We obtain this operator by inserting the analytical expression for the expansion coefficients [Eq. (3.5)] into the truncated expansion [Eq. (3.6)]. The best approximation operator is to be approximated by an operator of the form

$$\mathcal{P}_{(\boldsymbol{x},A)} : \mathbb{U} \to \mathbb{P} : u \mapsto \Psi A \mathcal{E}_{\boldsymbol{x}} u .$$

We regard the nodes $\boldsymbol{x}$ and the matrix A as parameters of this operator. Note that the output of these two operators are functions and not scalars as in the integration and quadrature operator [Eqs. (4.1) and (4.2)]. To measure the difference between $\mathcal{P}^\star$ and $\mathcal{P}_{(\boldsymbol{x},A)}$, I suggest two possible error measures.

1. The first one is similar to the OSC approach. We approximate $\mathcal{P}^\star$ by minimizing the operator norm of $\mathcal{P}^\star - \mathcal{P}_{(\boldsymbol{x},A)}$. Like in the OSC approach, the operators might be unbounded, so that the operator norm is not defined on $\mathbb{U}$. Therefore, we introduce a finite-dimensional test space $\mathbb{T}$ and the objective function becomes

$$\left(\boldsymbol{x}^\star, A^\star\right) = \underset{\substack{\boldsymbol{x}\in\Omega^n \\ A\in\mathbb{R}^{n\times n}}}{\operatorname{argmin}} \left\| \mathcal{P}^\star - \mathcal{P}_{(\boldsymbol{x},A)} \right\|_{\mathfrak{L}(\mathbb{T},\mathbb{P})} .$$

2. The second possible error measure is the Lebesgue constant:

$$\left(\boldsymbol{x}^\star, A^\star\right) = \underset{\substack{\boldsymbol{x}\in\Omega^n \\ A\in\mathbb{R}^{n\times n}}}{\operatorname{argmin}} \Lambda\left(\mathcal{P}_{(\boldsymbol{x},A)}\right)$$

   For the Lebesgue constant to be defined, we again have to restrict it to a test space $\mathbb{T}$, so the Lebesgue constant is the smallest number

$\Lambda\left(\mathcal{P}_{(\boldsymbol{x},A)}\right)$ satisfying

$$\left\|u - \mathcal{P}_{(\boldsymbol{x},A)}u\right\| \leq \left(1 + \Lambda\left(\mathcal{P}_{(\boldsymbol{x},A)}\right)\right)\left\|u - \mathcal{P}^{\star}u\right\|$$

for all $u \in \mathbb{T}$.

These two error measures are, of course, not the only possible ones. I selected these two specifically because both the operator norm and the Lebesgue constant are well known concepts in functional analysis and approximation theory, see Section 3.1.3.

On the one hand, this generalization of the OSC method can potentially lead to better sampling rules: the search space is larger than the one in the OSC method (from optimizing nodes and weights to optimizing nodes and the calculation rule) and the objective function is more closely adapted to the task of finding a response surface method. On the other hand, the optimization itself becomes more difficult: the output of the involved operators are functions, not scalars, so the evaluation of the objective function itself is more involved than for the OSC method. Furthermore, the dimension of the optimization problem is increased from $n(d+1)$ to $n(d+p)$, where n is the number of nodes, p is the number of expansion terms and d is the dimension of the parameter domain.

Whether the potential benefits are worth the additional optimization effort remains an open question for future research. I was able to obtain some preliminary results for the Lebesgue constant. These are presented in Appendix A.

4.5. Conclusions

In this chapter, I presented OSC, a sampling rule for the efficient, nonintrusive construction of polynomial response surfaces. The method is adapted to the task of calculating PCE coefficients and is formulated as an optimization problem. The nodes can be used with the quadrature approach or the least-squares approach and are not generally of tensor grid structure. By reusing nodes from previous calculations, nested integration rules can be constructed.

Numerical examples demonstrated that OSC sampling rules are stable, flexible and versatile. Stability means that nodes are distributed such that oscillations between the nodes (according to the Runge phenomenon) are avoided. Flexibility means that the number of nodes can be chosen relatively freely. Versatility means that input parameters with statistical dependencies can be handled directly, without transforming the input parameters to statistically independent parameters. With these properties, OSC showed to be the most efficient sampling rule among the ones compared in the numerical experiments.

Being formulated as an optimization problem, OSC depends on the availability of efficient optimization algorithms. In high dimensions, the optimization for finding the integration rule can take a considerable amount of time itself. Therefore, the OSC method is best applicable with parameter domains of dimension 10 or less. In higher-dimensional cases, the optimization time might become comparable to the computing time of the model function and then it might be overall more beneficial to use a non-optimal sampling method.

Last, I briefly outlined two variants of the OSC method that are more closely adapted to the task of building a polynomial response surface, but at the same time are more difficult to implement. A detailed analysis of these variants is an open topic for future research.

5. Towards Optimal Response Surface Modeling

In this chapter, we attempt to find an optimal response surface method for uncertainty propagation. While in the previous chapter, we only optimized the sampling rule for response surfaces of a certain type (namely polynomials), we now consider the optimization over all nonintrusive, linear response surface methods. This means that we optimize both the sampling rule and the functional form of the response surface. That way, we allow the ansatz space $\mathbb{P}$ to be chosen arbitrarily. The goal of this generalization is to determine the ansatz space via optimization, so that the practitioner does not have to choose it.

Central to any optimization is the objective function. The main concern of this chapter is the question of how to define an appropriate objective function that measures how good a response surface method is. I expect three benefits from defining an objective function. First, once two practitioners have agreed on one objective function, then this solves any possible dispute over which method is better. Second, with an objective function in place, we can apply formal optimization methods to find the best possible method. Third, in formulating an objective function we are forced to state all of our assumptions explicitly. Depending on the type of objective function, such assumptions could come in form of a test space which is assumed to contain the model function, or in form of a random field from which the model function is assumed to be a realization. Making all assumptions explicit helps gain a better understanding of when to apply what method.

In the following section, I propose three possible objective functions, discuss them in terms of practicality and, if possible, derive their optimal method. Afterwards, I provide mathematical proofs and finish the chapter with a conclusion.

5.1. Objective Functions

We start with the general formulation given in Section 3.1.1. Any non-intrusive, linear response surface methods can be written as an operator $\mathcal{P} : \mathbb{U} \to \mathbb{U}$, with $\mathcal{P} = \Phi\mathcal{E}_{\boldsymbol{x}}$, consisting of an evaluation operator $\mathcal{E}_{\boldsymbol{x}}$ and an operator Φ mapping the model response back to a function. The space of all such operators with n function evaluations is called $\mathfrak{P}_n$. It is parametrized by n nodes $\boldsymbol{x} = (x_1, \ldots, x_n)$ and n unit response functions $\Phi_1, \ldots, \Phi_n$.

Given the input parameters X and the model function u, we obtain $Y = u(X)$. Applying the response surface operator, we get $\tilde{u} = \mathcal{P}u$ and then $\tilde{Y} := \tilde{u}(X)$. The goal is to minimize the L^2-error between the model function and the response surface, see Section 2.1.3:

$$\begin{aligned} \left\|Y - \tilde{Y}\right\| &= \|u - \tilde{u}\| \\ &= \|\mathcal{I}u - \mathcal{P}u\| \\ &= \|(\mathcal{I} - \mathcal{P})\, u\|, \end{aligned}$$

where $\mathcal{I}$ is the identity operator on $\mathbb{U}$. The operator $(\mathcal{I} - \mathcal{P})$ maps a function u to its total error $u - \tilde{u}$.

The L^2-error $\|(\mathcal{I} - \mathcal{P})\, u\|$ itself is not useful as an objective function because it depends on the unknown or uncertain model function u. So instead we design the operator $\mathcal{P}$ such that the error $\|(\mathcal{I} - \mathcal{P})\, u\|$ is small for a class of functions. The purpose of the objective function is to formally define what is meant by "a small error on a class of functions". In the following sections, I examine three possible objective functions: the Lebesgue constant, the operator norm, and the expected squared error. All three objective functions are centered around the minimization of the L^2-error, but differ in the way they consider different possible shapes of u.

The discussion of the Lebesgue constant is very short because as it turns out the Lebesgue constant is not applicable for an optimization over $\mathfrak{P}_n$. For each of the other two objective functions, I discuss two aspects:

Optimal Operator Is there an optimal response surface operator with respect to the objective function?

Practicality How useful is the objective function in practice? How realistic are the underlying assumptions?

5.1.1. Lebesgue Constant

The first objective function is the Lebesgue constant $\Lambda(\mathcal{P})$. Recall that the Lebesgue constant relates the achieved error $\|u - \mathcal{P}u\|$ with the best possible error $\|u - u^\star\|$ in a certain space $u^\star \in \mathbb{P}$. It is the smallest number $\Lambda(\mathcal{P})$, such that for all $u \in \mathbb{U}$ it holds

$$\|u - \mathcal{P}u\| \leq (1 + \Lambda(\mathcal{P})) \|u - u^\star\|,$$

see Section 3.1.3. In the formulation that I provided in that section, the reference space $\mathbb{P}$ is defined as the image of the considered operator: $\mathbb{P} = \mathrm{Im}(\mathcal{P})$.

Now each operator in $\mathfrak{P}_n$ may have a different image. Therefore, different operators relate their achieved error $\|u - \mathcal{P}u\|$ to different reference errors $\|u - u^\star\|$, which makes the comparison of the Lebesgue constant of operators with different images meaningless. As an extreme example, consider the null operator: for any model function u, it follows $\mathcal{P}u = 0$ and also the best approximation in the image of the null operator is $u^\star = 0$. Therefore, the null operator achieves a Lebesgue constant of 0 (which is the best possible value), even though it clearly does not yield a useful response surface.

This example shows that it only makes sense to compare operators in terms of their Lebesgue constant if the Lebesgue constant is calculated with respect to the *same* reference space $\mathbb{P}$. The only reference space that contains the images of all operators in $\mathfrak{P}_n$ is $\mathbb{U}$ itself. The choice $\mathbb{P} := \mathbb{U}$, however, does not lead to a useful Lebesgue constant either because the best approximation of any model function u in $\mathbb{U}$ is $u^\star = u$ itself, so the reference error follows as $\|u - u^\star\| = 0$ and the right hand side in the definition of $\Lambda(\mathcal{P})$ vanishes. The Lebesgue constant can only be finite if also the achieved error $\|u - \mathcal{P}u\|$ on the left hand side is zero. This is only the case for the identity operator, which generally does not lie in $\mathfrak{P}_n$.

I conclude that the Lebesgue constant is not applicable as an objective function if we want to optimize over the space $\mathfrak{P}_n$ because it does not help us identify good unit response functions. If, however, the ansatz space is fixed beforehand, then the Lebesgue constant can be a useful objective function for finding optimal nodes $\boldsymbol{x}$ as I demonstrated in Section 4.4 in the previous chapter.

5.1.2. Operator Norm

The second objective function is the operator norm of $(\mathcal{I} - \mathcal{P})$, see Section 3.1.3. The operator norm $\|\mathcal{I} - \mathcal{P}\|_{\mathfrak{L}(\mathbb{U},\mathbb{U})}$ over the whole space $\mathbb{U}$ is undefined because operator $\mathcal{P}$ involves the evaluation operator $\mathcal{E}_{\boldsymbol{x}}$, which is unbounded. Therefore, similarly to the procedure in Chapter 4, we introduce a test space $\mathbb{T} \subset \mathbb{U}$, such that $\|\mathcal{I} - \mathcal{P}\|_{\mathfrak{L}(\mathbb{T},\mathbb{U})}$ is defined for at least one non-trivial choice of $\mathcal{P}$. This can, for example, be achieved by making $\mathbb{T}$ finite-dimensional. The objective function then reads

$$f(\mathcal{P}) := \|\mathcal{I} - \mathcal{P}\|_{\mathfrak{L}(\mathbb{T},\mathbb{U})}\,.$$

Recall that the operator norm is defined via a supremum over the elements of $\mathbb{T}$. Therefore, this objective function can be thought of as a worst-case error, similar to the ∞-norm for functions or sequences.

For this objective function, the optimal operator is not difficult to obtain. However, as I will discuss in the practicality section below, this approach is unpractical in most cases.

The Optimal Operator

The optimal operator with respect to the first objective function is either the identity operator or the null operator, depending on the dimensionality of space $\mathbb{T}$ as stated in the following theorem.

Theorem 1

Let $\mathbb{T}$ be a subspace of $\mathbb{U}$ and define $t := \dim \mathbb{T}$ (possibly including the case $t = \infty$).

Then, the minimum of the objective function $f(\mathcal{P}) = \|\mathcal{I} - \mathcal{P}\|_{\mathfrak{L}(\mathbb{T},\mathbb{U})}$ over $\mathfrak{P}_n$ is

$$\min_{\mathcal{P}\in\mathfrak{P}_n} \|\mathcal{I} - \mathcal{P}\|_{\mathfrak{L}(\mathbb{T},\mathbb{U})} = \begin{cases} 0 & \text{if} \quad t \leq n, \\ 1 & \text{otherwise.} \end{cases}$$

This minimum is achieved by operator $\mathcal{P}^\star$ with

$$\mathcal{P}^\star = \begin{cases} \mathcal{I} & \text{if } t \leq n \\ 0 & \text{otherwise.} \end{cases}$$

A proof is provided separately in Section 5.2.1.

If the number of nodes n is larger or equal to the dimension of $\mathbb{T}$, then we are able to identify all elements of $\mathbb{T}$ and therefore we can reconstruct all elements with zero error. If, however, the dimension of $\mathbb{T}$ is larger than n, then there will always be at least one non-zero element in $\mathbb{T}$ that is mapped to zero, so with respect to the worse-case objective function, no operator can be better than the null operator.

Practicality

The main use of an operator norm is its submultiplicativity property, which yields an upper bound for the total error [Eq. (3.17)]. For all $u \in \mathbb{T}$, it is

$$\|u - \tilde{u}\| \leq \|\mathcal{I} - \mathcal{P}\|_{\mathfrak{L}(\mathbb{T},\mathbb{U})} \|u\| .$$

For this bound to be useful, two conditions have to be satisfied. First, the test space must be chosen such that it contains the model function. Otherwise the upper bound does not hold. Second, the operator norm must be smaller than 1. Otherwise the total error can be as large as the model function itself.

According to the theorem, the second condition can only be met if $\mathbb{T}$ is finite dimensional. So for this approach to be useful, we need to know a finite-dimensional space that is guaranteed to contain the model function. This is a requirement which I believe is not met in most applications. Overall, I conclude that this objective function does not lead to a useful response surface method.

This analysis reveals two more interesting aspects:

First, all that matters about the test space $\mathbb{T}$ is its dimension and not the smoothness of its elements. As the theorem shows, the smoothness of the model function does not matter if we consider a worst-case error. In the following section we will see that this is different if we consider an average error instead.

Second, if the requirements of the theorem are met and the dimension of $\mathbb{T}$ is smaller than n, then the actual placement of the nodes is almost irrelevant. As an example, if $\mathbb{T}$ is a space of polynomials up to degree $n - 1$, then any set of n distinct nodes allows us to construct the identity operator. Of course, different node sets might lead to a more or less stable reconstruction, but the operator norm objective function does not capture the aspect of stability.

5.1.3. Expected Squared Error

The third objective function is the average squared error over a set of functions, similarly to a 2-norm error. For the notion of *average over a set of functions*, we need to equip the set of functions with a measure. In other words, we have to describe the possible shapes of the model function u by a random field U. A similar approach has been used previously to find optimal approximations for functionals such as integration [98, 108, 114].

Let U be a random field over Ω, with $U \in \mathbb{U}$ almost surely. The objective function for an operator $\mathcal{P}$ is then

$$f(\mathcal{P}) := \mathrm{E}_U\left[\|(\mathcal{I} - \mathcal{P})U\|^2\right].$$

For each realization u of U, $(\mathcal{I} - \mathcal{P})u$ is an element of $\mathbb{U}$, so the norm $\|\cdot\|$ in this definition is the L^2-norm.

The Optimal Operator

Finding the optimal operator with respect to the second objective function is a bit more involved than for the previous objective function. The analysis is split into two parts. First, I determine the optimal unit response functions $\Phi_1, \ldots, \Phi_n$ under the assumption that the nodes $x_1, \ldots, x_n$ are given. Then, in the second step, I set up a criterion for the optimality of the nodes themselves.

Optimal Unit Response Functions for Given Nodes

At first, we derive the optimal set of unit response functions $\Phi_1, \ldots, \Phi_n$ for the case that the nodes $x_1, \ldots, x_n$ are given. The following results and derivations are well known in the context of Kriging and geostatistics [77]. For fixed nodes, we can explicitly solve for the optimal operator $\mathcal{P}$:

Theorem 2

Let U be a random field over Ω, with known and finite raw second moments function $C(x, x') = \mathrm{E}_U[U(x)U(x')]$ for any $x, x' \in \Omega$. Let $\boldsymbol{x} = [x_1, \ldots, x_n] \in \Omega^n$ be a vector of nodes such that the $n \times n$ matrix Q with entries $[Q]_{ij} = C(x_i, x_j)$ is non-singular.

Then, there is an optimal operator of the form $\mathcal{P}^\star = \Phi^\star \mathcal{E}_{\boldsymbol{x}}$ that minimizes the objective function $f(\mathcal{P}) = \mathrm{E}_U\left[\|(\mathcal{I} - \mathcal{P})U\|^2\right]$ over $\mathfrak{P}_n$. The unit response *functions $\Phi^\star$ of the optimal operator satisfy*

$$\Phi^\star(x) = q(x)\,Q^{-1}, \tag{5.1}$$

where $q(x) = [q_1(x), \dots, q_n(x)]$ is the vector containing the functions $q_i : \Omega \to \mathbb{R} : x \mapsto C(x_i, x)$.

Furthermore, the minimal pointwise expected squared error for a parameter value $x \in \Omega$ is

$$\mathrm{E}_U\left[\left(\left(\mathcal{I} - \mathcal{P}^\star\right)U\right)(x)^2\right] = C(x,x) - q(x)\,Q^{-1}q(x)^\top, \tag{5.2}$$

and the minimal value in the objective function is

$$\mathrm{E}_U\left[\left\|\left(\mathcal{I} - \mathcal{P}^\star\right)U\right\|^2\right] = \mathrm{E}\left[\|U\|^2\right] - \int_\Omega q(x)\,Q^{-1}q(x)^\top \,\mathrm{d}\mu(x). \tag{5.3}$$

The proof is given separately in Section 5.2.2. Note, that this theorem does not assume U to be a Gaussian random field. It only requires a finite second moment function.

The result in Eq. (5.1) shows that the optimal response surface is the unique linear combination of the functions q_i that interpolates u in the nodes. The resulting response surface model $\mathcal{P}^\star u$ is the Kriging estimator and the expected squared pointwise error [Eq. (5.2)] is known from Kriging in exactly this form, see Eq. (3.19) in Section 3.2. In the literature, the Kriging estimator is known as the best linear unbiased estimator (BLUE) [77], so Theorem 2 is simply a restatement of the fact that the optimal response surface method is the best linear unbiased estimator. But Theorem 2 is interesting for a different reason: it reveals a connection between response surface models and random fields. Earlier, in Chapter 3, I introduced these two concepts as two different tools. Now it turns out that under certain conditions, the mean of a random field is the best possible response surface.

Note that for fixed nodes, the optimal operator in Eq. (5.1) does not depend on the measure μ of the parameters, but only on the random field U, namely on the second moment function C. The same holds for the pointwise expected squared error [Eq. (5.2)]. Only the expected squared L^2-error [Eq. (5.3)] depends on μ.

It is worth pointing out one difference to the common Kriging methodology: by forcing $\mathcal{P}$ to be a *linear* operator, the method is centered around zero. Therefore, we have to consider the raw second moments instead of the central second moments (covariances). If the model has a known mean $m(x) = \mathrm{E}_U[U(x)]$, then one would rather consider *affine* operators of the form $\mathcal{P} = \Phi\mathcal{E}_{\boldsymbol{x}} + T$. Inserting this approach into the same optimality condition, one finds that the optimal operator satisfies $\mathcal{P}u = \Phi\mathcal{E}_{\boldsymbol{x}}(u - m) + m$. The basis functions Φ would still satisfy the condition in Eq. (5.1), only that the raw second moment function C would be replaced by the covariance function $C(x, x') = \mathrm{Cov}[U(x), U(x')]$. This approach would lower the expected error in Eq. (5.3) even further. This demonstrates that choosing a linear operator for $\mathcal{P}$ is only appropriate, when U is assumed to have a zero mean. Details on Kriging and a derivation are provided, for example, by Kitanidis [77].

Finding Optimal Nodes

In the expression for the expected error in Eq. (5.3), the first term on the right-hand side is independent from the nodes, so finding optimal nodes is a matter of maximizing the last term:

$$\int_\Omega q(x)\, Q^{-1} q(x)^\top \,\mathrm{d}\mu(x)\,.$$

This problem is known in the field of geostatistical optimal design of experiments. The criterion is called *integrated mean squared error* [99, 128]. The only difference from optimal design for standard Kriging methods is that the integral over the parameter domain is weighted via the measure μ, whereas standard Kriging methods consider a uniformly weighted integral. If μ is a discrete measure, then the optimality criterion reduces to the *A-criterion* [99, 104].

The further derivation of specific criteria for the optimality of a set of nodes is problem dependent and out of the scope of this thesis. For some choices of random fields U and underlying measure μ, the optimal nodes can be found analytically [4]. Otherwise, one can try to find nodes via numerical optimization.

Practicality

The role of specifying the random field U is the same as modeling the input parameters by a random variable X. In Section 2.1.1, where I introduced X as the model parameters, I already mentioned that it is a strong assumption to assume that we can specify a distribution for X. The same holds for the random field U. However, I believe it is a much weaker assumption than the one in the previous section, where the practitioner is required to specify a finite-dimensional space $\mathbb{T}$ that contains u.

The meaning of the random field U can be interpreted in two different ways.

- In the first interpretation, the random field is an expression of the belief about the model function u in the Bayesian sense. In this interpretation, the practitioner has a belief about the model function anyways, she just needs to write it down as a random field. The resulting response surface method is then the optimal one according to the practitioners belief: if someone believes in random field U, then it is only rational to use the corresponding optimal response surface method. There is no guarantee, however, that the resulting surrogate $\mathcal{P}u$ is close to u because the belief might be wrong in the first place.
- According to the second interpretation, specifying U is a matter of modeling. After all, the main downside of the Bayesian approach is that it is not fully clear what is meant by "just writing down ones belief as a random field". Labeling the choice of U as a modeling question and giving some practical guidelines might overall be more useful than requiring the practitioner to express her belief in form of a random field.

As a practical aid, let me briefly outline two possible modeling approaches for setting up a random field U according to the second interpretation.

First, it is possible to model U as a stationary Gaussian random field. Such field is completely described by the field mean m and covariance function C, see also Section 3.2. As I already mentioned, the covariance function can express a smoothness assumption about model function. For example, the Matérn covariance function has a smoothness parameter that is directly related to how often the realizations of the random field are differentiable [42, 43, 63].

Second, any model function in L^2 can be expanded via its PCE (see Section 3.1.2)

$$u = \sum_{i=1}^{\infty} c_i \Psi_i .$$

Since u has a finite variance, the sequence of coefficients c_i converges to zero. The convergence rate depends on the smoothness of u [96]. This motivates to model the random field as

$$U = \sum_{i=1}^{\infty} \lambda_i X_i \Psi_i .$$

Here, $\lambda_i \in \mathbb{R}^+$ is a sequence converging to zero at a certain rate based on the knowledge about u. The $X_i \sim \mathcal{N}(0,1)$ are i.i.d. normally distributed random variables. This field U is a Gaussian random field, but it is not stationary. The advantage of this type of random field is that it clearly and explicitly reflects a smoothness assumption about the model function and allows the model function to have a trend. A disadvantage is, that it introduces a conceptual dependency between the measure μ and the field U because the basis polynomials Ψ_i are orthonormalized with respect to μ.

Let us return to the discussion of the practicality of describing the model function by a random field. Once we accept the assumption that the model function is a realization from a given random field, the optimal operator according to Theorem 2 has a number of convenient properties.

- The optimal response surface $\mathcal{P}u$ coincides with the Kriging mean. Kriging is a well-known and well-understood method and software solutions are easily available to practitioners.
- The random field U is not required to be a Gaussian random field. We can equally well handle non-Gaussian random fields. Such fields would occur, for example, if the model function is known to be discontinuous with an uncertain jump location.
- If the field U is Gaussian, then the pointwise expected squared error $\mathrm{E}_U\left[\left(\left(\mathcal{I} - \mathcal{P}^{\star}\right) U\right)(z)^2\right]$ coincides with the posterior estimation variance of field U conditioned to the model responses. It quantifies the residual uncertainty in the response surface $\mathcal{P}u$. Previously, I was referring to this as the *accuracy* of the response surface, but since this approach is Bayesian, I would rather call it *certainty* or *uncertainty*, more precisely, code uncertainty. Together with the expected error, the optimal response surface becomes an emulator. As presented in Chapter 3, an emulator contains more information than a response surface.
- The response surface model $\mathcal{P}u$ is optimized in a pointwise way: the value of $\mathcal{P}u(x)$ was chosen optimally for each $x \in \Omega$ individually. As a result, the optimal response surface is independent of the measure μ. In stochastic collocation methods, the functional form of the

response surface is restricted to a certain class of functions (to polynomials). Such a restriction necessarily leads to local errors between the surrogate and the model function. The measure μ dictates how the local errors are distributed over the domain. So in stochastic collocation, the response surface model $\mathcal{P}u$ is influenced by the measure μ. As I discussed in Section 2.1.1, it is desirable to not couple the surrogate model with the measure.

One last important aspect is that the second-moment function C directly influences how good a response surface method can be. This can easily be understood with the two possible extreme cases. The one extreme case is when the second-moment for all points is constant $C(x, x') = \text{const}$. Then, one model evaluation is enough to reduce the expected squared error to zero (see Eq. (5.3)) because each realization of the random field U would be a constant function. The other extreme case is when the model response at all points is completely uncorrelated: $C(x, x') = \text{const} \cdot \delta_{xx'}$. Then, a point evaluation does not give any information about the model response at other points and so the expected squared error can only be reduced if the underlying measure μ is a discrete measure (and if the nodes lie exactly on the discrete points of the measure). If the measure is continuous, then the expected squared error cannot be reduced at all. To summarize, the stronger the random field is correlated, the easier it is to design good response surfaces. The smoothness of the random field is closely related to the correlation structure, so if the model function is known to be smooth, then one can generally expect it to be well approximated using a response surface method.

Figure 5.1 shows this relation schematically. On the left, a stationary prior field is shown with its variance (gray shaded area). The center plot and the right plot show two different conditioned fields, both with an Gaussian covariance structure, $C(x, x') = \exp\left[-\left(x - x'\right)^2 / l^2\right]$, but with different correlation lengths. The center plot has a correlation length of one-tenth of the parameter domain, while the right plot has a correlation length of half of the domain.

This relation between the smoothness of a random field and the performance of a response surface method is conditional to the fact that the model function is a realization of the random field. This is important. Otherwise one might by mistake come to the conclusion that it is always better to select highly smooth random fields. This conclusion is incorrect: if the random field is very smooth, but the model function is not, then the resulting

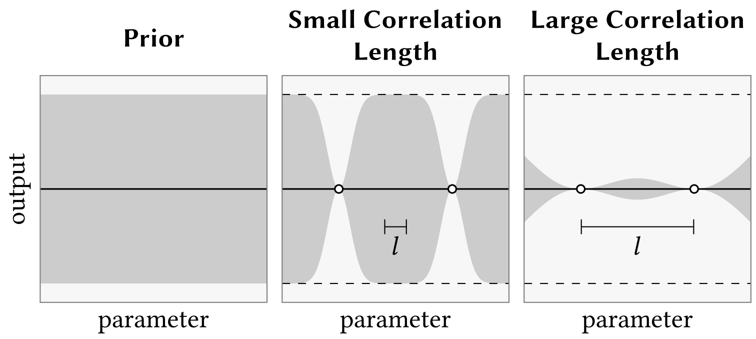

Figure 5.1.: Influence of the correlation function on the residual code uncertainty (shown as gray shaded area). Left: situation before any model evaluations. Center and right: situation after two model evaluations (circles) with different correlation lengths.

response surface will be overconfident and possibly misleading.

5.2. Proofs

This section contains the proofs of Theorem 1 and Theorem 2. The proofs are provided for completeness and are not required for the discussion at the end of the chapter. A reader who is mainly interested in the results and the discussion may safely skip this section and proceed with the conclusions in Section 5.3.

Unless noted otherwise, I developed these proofs on my own. I am fairly sure, though, that similar theorem already exist in other fields.

5.2.1. Theorem 1

We begin with the following auxiliary lemma which is used in the proof of Theorem 1.

Lemma

Let $f_1, \ldots, f_n$ be linearly independent functions. Then there exist points $x_1, \ldots, x_n \in \Omega$, such that the $n \times n$ matrix

$$A_n = \begin{bmatrix} f_1(x_1) & \ldots & f_1(x_n) \\ \vdots & & \vdots \\ f_n(x_1) & \ldots & f_n(x_n) \end{bmatrix}$$

is non-singular.

I developed the following proof on my own, but I would not be surprised if this or similar lemmas have already been proved in the mathematical literature.[1]

Proof of Lemma

We prove by induction.

For $n = 1$, we select any point x_1, such that $f_1(x_1) \neq 0$. Such point exists because, otherwise, $f_1 = 0$, which is linearly dependent.

Now assume that there exist points $x_1, \ldots, x_n$, such that A_n is non-singular. Define the row vectors $v_i := [f_i(x_1), \ldots, f_i(x_n)]$ for $i = 1, \ldots, n+1$. The first n vectors $v_1, \ldots, v_n$ form a basis of $\mathbb{R}^n$ and we can expand the $n+1$-th vector as

$$v_{n+1} = \sum_{i=1}^{n} a_i v_i.$$

The basis functions f_i are linearly independent, and so $r_{n+1} := f_{n+1} - \sum_{i=1}^{n} a_i f_i \neq 0$. Therefore, there exists a point x_{n+1}, such that $r_{n+1}(x_{n+1}) \neq 0$. Inserting this point x_{n+1} into A_{n+1}, we may replace the last row $[f_{n+1}(x_1), \ldots, f_{n+1}(x_{n+1})]$ by the row $[r_{n+1}(x_1), \ldots, r_{n+1}(x_{n+1})] = [0, \ldots, 0, r_{n+1}(x_{n+1})]$ without changing the determinant of A_{n+1}. Developing the determinant of A_{n+1} with respect to the last row yields $\det A_{n+1} = r_{n+1}(x_{n+1}) \cdot \det A_n \neq 0$ and, therefore, A_{n+1} is non-singular. $\square$

Now we come to the proof of Theorem 1.

[1] I asked several mathematicians, where in the literature I could find a proof for this or a similar lemma. One of them, Prof. (em.) Dr. Robert Schaback, kindly replied that a proof by induction would be so simple that developing an own proof would be faster than finding a literature reference.

Proof of Theorem 1

In the first part of this proof, we show that the minimal operator norm is zero for $t \leq n$ and then, in the second part, we prove that it is 1 otherwise.

1. It is already possible to achieve an operator norm of zero with $t = n$. If n is even larger, then any further nodes can be chosen arbitrarily. To prove that the minimal operator norm is zero if $t = n$, let $F = [f_1, \dots, f_t]$, such that the functions $f_1, \dots, f_t$ form a basis of $\mathbb{T}$. We now construct the operator $\mathcal{P}$ such that $\mathcal{I} = \mathcal{P}$ on $\mathbb{T}$.
   Let $u \in \mathbb{T}$ be any function. We can expand it as $u = \sum_{i=1}^{t} c_i f_i = Fc$, with a column vector $c = [c_1, \dots, c_t]^\top$. Select nodes $\boldsymbol{x} = [x_1, \dots, x_t] \in \Omega^t$ such that the $t \times t$ matrix A with $[A]_{ij} = f_i\left(x_j\right)$ is non-singular, according to the previous lemma. Inserting these points into u we obtain for any $j = 1, \dots, t$: $u\left(x_j\right) = \sum_{i=1}^{t} c_i f_i\left(x_j\right)$; in vector-matrix notation this is $\mathcal{E}_{\boldsymbol{x}} u = Ac$, where $\mathcal{E}_{\boldsymbol{x}} u = [u(x_1), \dots, u(x_t)]^\top$ and $c = [c_1, \dots, c_t]^\top$. Since A is non-singular, we can rearrange: $c = A^{-1}\mathcal{E}_{\boldsymbol{x}} u$. Now

$$\begin{aligned} u &= Fc \\ &= FA^{-1}\mathcal{E}_{\boldsymbol{x}} u \\ &= [\Phi_1, \Phi_2, \dots, \Phi_t]\, \mathcal{E}_{\boldsymbol{x}} u \\ &= \mathcal{P} u, \end{aligned}$$

   where we define the functions Φ_i via $[\Phi_1, \dots, \Phi_t] := FA^{-1}$, and then $\mathcal{P} = [\Phi_1, \dots, \Phi_t]\, \mathcal{E}_{\boldsymbol{x}}$. It is $\mathcal{P}u = u = \mathcal{I}u$ for all $u \in \mathbb{T}$, and so the operator norm of $\mathcal{I} - \mathcal{P}$ is zero.

2. We now prove the second part of the theorem: if $t > n$, then the minimal operator-norm is 1. First, I show that the minimal operator-norm cannot be smaller than 1 and then I show that the value of 1 can be achieved.

   Assume that $t > n$. The image of $\mathcal{P}$ is the span of the unit response functions $\operatorname{Im}(\mathcal{P}) = \operatorname{span}\{\Phi_1, \dots, \Phi_n\}$, therefore for the dimension of the image it follows $\dim[\operatorname{Im}(\mathcal{P})] \leq n$. According to the rank-nullity-theorem [88], $\dim \mathbb{T} = \dim[\operatorname{Im}(\mathcal{P})] + \dim[\operatorname{Ker}(\mathcal{P})]$, it follows that $\dim[\operatorname{Ker}(\mathcal{P})] = \dim \mathbb{T} - \dim[\operatorname{Im}(\mathcal{P})] > 0$. So the kernel of $\mathcal{P}$ contains at least one non-trivial function u, and it holds $\mathcal{P}u = 0$. Applying the operator leads to $\|(\mathcal{I} - \mathcal{P})u\| / \|u\| = \|u\| / \|u\| = 1$. This constitutes a lower bound for the operator norm of $(\mathcal{I} - \mathcal{P})$.

The minimal value of 1 is achieved by the null-operator. By setting $\mathcal{P} = 0$, we obtain $\|\mathcal{I} - \mathcal{P}\|_{\mathfrak{L}(\mathbb{T},\mathbb{U})} = \|\mathcal{I}\|_{\mathfrak{L}(\mathbb{T},\mathbb{U})} = 1$.

This concludes the proof. $\square$

5.2.2. Theorem 2

This proof is heavily based on the derivations by Kitanidis [77]. Only two aspects are changed from the original work: first, the notation is adapted, and second, in this proof, the unbiasedness condition from Kriging is not needed, so that the final result has a slightly different form.

Proof of Theorem 2

We construct the optimal operator by pointwise optimization. If an operator $\mathcal{P}$ minimizes

$$\mathrm{E}_U \left[\left(\left(\mathcal{I} - \mathcal{P} \right) U \right) (x)^2 \right] \tag{5.4}$$

in each point $x \in \Omega$ and lies in the search space by having the form $\mathcal{P} = \Phi \mathcal{E}_{\boldsymbol{x}}$, then it also minimizes $\mathrm{E}_U \left[\|(I - \mathcal{P}) U\|^2 \right]$ because

$$\begin{aligned} \mathrm{E}_U \left[\|(\mathcal{I} - \mathcal{P}) U\|^2 \right] &= \mathrm{E}_U \left[\int_\Omega \left(\left(\mathcal{I} - \mathcal{P} \right) U \right) (x)^2 \, \mathrm{d}\mu (x) \right] \\ &= \int_\Omega \underbrace{\mathrm{E}_U \left[\left(\left(\mathcal{I} - \mathcal{P} \right) U \right) (x)^2 \right]}_{\text{pointwise minimal}} \mathrm{d}\mu (x) . \end{aligned} \tag{5.5}$$

We now insert the form $\mathcal{P} = \Phi \mathcal{E}_{\boldsymbol{x}}$ into the pointwise minimization in Eq. (5.4). The inner part $\mathcal{P} U (x)$ becomes

$$\begin{aligned} \mathcal{P} U (x) &= \left[\Phi_1 (x), \ldots, \Phi_n (x) \right] \mathcal{E}_{\boldsymbol{x}} U \\ &= \sum_{i=1}^{n} \Phi_i (x) U (x_i) . \end{aligned}$$

It follows

$$\begin{aligned}
\left(\left(\mathcal{I}-\mathcal{P}\right)U\right)(x)^2 &= \left(U(x)-\sum_{i=1}^{n}\Phi_i(x)\,U(x_i)\right)^2 \\
&= U(x)^2 - 2\sum_{i=1}^{n}\Phi_i(x)\,U(x_i)\,U(x) \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{n}\Phi_i(x)\,\Phi_j(x)\,U(x_i)\,U\left(x_j\right).
\end{aligned}$$

The expected value with respect to U is

$$\begin{aligned}
\mathrm{E}_U\left[\left(\left(\mathcal{I}-\mathcal{P}\right)U\right)(x)^2\right] &= C(x,x) - 2\sum_{i=1}^{n}\Phi_i(x)\,C(x,x_i) \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{n}\Phi_i(x)\,\Phi_j(x)\,C\left(x_i,x_j\right) \qquad (5.6)
\end{aligned}$$

We derive this expression with respect to $\Phi_k(x)$

$$\frac{\partial \mathrm{E}_U\left[\left(\left(\mathcal{I}-\mathcal{P}\right)U\right)(x)^2\right]}{\partial \Phi_k(x)} = -2C(x,x_k) + 2\sum_{i=1}^{n}\Phi_i(x)\,C(x_i,x_k).$$

For finding the minimum with respect to $\Phi_k(x)$, we set the derivative to zero and obtain a system of equations for the optimal unit response functions $\Phi^\star$:

$$\sum_{i=1}^{n}\Phi_i^\star(x)\,C(x_i,x_k) = C(x,x_k).$$

In vector notation with $[Q]_{ij} = C\left(x_i,x_j\right)$ and $q(x) = [C(x,x_1),\ldots,C(x,x_n)]$, we get

$$\begin{aligned}
\left[\Phi_1^\star(x),\ldots,\Phi_n^\star(x)\right]Q &= q(x) \\
\left[\Phi_1^\star(x),\ldots,\Phi_n^\star(x)\right] &= q(x)\,Q^{-1} \qquad (5.7) \\
\Phi^\star(x) &= q(x)\,Q^{-1}.
\end{aligned}$$

The matrix Q is invertible by assumption.

For the pointwise error, we insert Eq. (5.7) into Eq. (5.6) and use the fact that Q^{-1} is symmetric:

$$\begin{aligned}
\mathrm{E}_U\left[\left(\left(\mathcal{I}-\mathcal{P}^\star\right)U\right)(x)^2\right] &= C(x,x) - 2\sum_{i=1}^{n}\Phi_i^\star(x)\,C(x,x_i) \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{n}\Phi_i^\star(x)\,\Phi_j^\star(x)\,C\left(x_i,x_j\right) \\
&= C(x,x) - 2\Phi^\star(x)\,q(x)^\top \\
&\quad +\Phi^\star(x)\,Q\Phi^\star(x)^\top \\
&= C(x,x) - 2q(x)\,Q^{-1}q(x)^\top \\
&\quad + \left(q(x)\,Q^{-1}\right)Q\left(Q^{-1}q(x)^\top\right) \\
&= C(x,x) - q(x)\,Q^{-1}q(x)\,.
\end{aligned}$$

Finally, using Eq. (5.5), we obtain

$$\begin{aligned}
\mathrm{E}_U\left[\left\|\left(\mathcal{I}-\mathcal{P}^\star\right)U\right\|^2\right] &= \int_\Omega \mathrm{E}_U\left[\left(\left(\mathcal{I}-\mathcal{P}^\star\right)U\right)(x)^2\right]\,\mathrm{d}\mu(x) \\
&= \int_\Omega C(x,x) - q(x)\,Q^{-1}q(x)^\top\,\mathrm{d}\mu(x) \\
&= \mathrm{E}_U\left[\|U\|^2\right] - \int_\Omega q(x)\,Q^{-1}q(x)^\top\,\mathrm{d}\mu(x)\,. \qquad \square
\end{aligned}$$

5.3. Conclusions

In this chapter, I applied the idea of optimality to the construction of response surface methods for uncertainty propagation. I compared and discussed three possible objective functions for measuring the quality of a method. If possible, I derived the optimal method according to these objective functions:

1. The first objective function, the Lebesgue constant, is not applicable for an optimization over all possible nonintrusive linear response surface methods because it requires the ansatz space to be fixed beforehand.
2. The second objective function, the operator norm, results in an optimal method with a very strong property: a response surface constructed by such method has a guaranteed error bound. However,

this objective function requires very strong assumptions about the model function: the user of the method has to be able to specify a finite-dimensional space that contains the model function. I consider this assumption too strong for practical purposes.

3. The third objective function, the expected squared error over a random field, leads to an optimal surrogate modeling method that coincides with Kriging. For this approach, the user is required to specify a random field from which the model function is a realization. This is a lower requirement than for the previous objective function, but in return the result is a bit weaker: the optimal method only provides an error bound for the *expected* total error. Furthermore, the correlation function C has direct influence on the expected squared error that can be achieved. A field with strong correlations (for example a very smooth field) generally leads to a better error than a field with almost no correlations, as shown in Figure 5.1 on page 90. This shows that if we make more assumptions about the model function and these assumptions are valid, then we are able to achieve better results.

This conclusion is very similar to the No Free Lunch (NFL) theorems for optimization. The NFL theorems state that on average over all possible optimization problems, all optimization algorithms perform equally well [164]. In consequence, an optimization algorithm can only perform better than a random search if additional knowledge about the problem at hand is available. The same holds for response surface methods: a method can only be better than the null-operator if additional knowledge about the model function is available.

6. Sequential Design of Computer Experiments for Bayesian Inverse Problems

In this chapter, we turn to Bayesian inverse problems. Under the assumption of an expensive model function, the overall goal is to solve Bayesian inverse problems with as few function evaluations as possible. Bayesian inverse problems differ from uncertainty propagation problems in one very important aspect: in Bayesian inverse problems, we do not know beforehand, which parts of the parameter domain are most important. This is a chicken and egg situation. The parts most important for Bayesian inverse problems are those with a large posterior probability mass, so if possible, we would like to sample the model function most densely there. However, to find those parts in the parameter domain, we need to sample the model function first.

A possible approach is to simply ignore the fact that different parts of the domain are more important than others, and therefore sample according to the prior distribution. If the prior is completely covered, then one would expect that also all relevant parts for the posterior are covered. This is indeed true: if a sequence of response surface models converges to the model function (with respect to the prior) and we insert these response surfaces into Bayes' theorem, then (under certain assumptions) the sequence of approximate posterior distributions converges to the true posterior [35, 96].

Following this approach, however, we miss the opportunity to learn about the model function after each function evaluation and therefore end up with a method that is convergent but not optimal. For this reason, I propose a sequential design of computer experiments. Recall that in the context of emulators, the selection of nodes is often called a "design", rather than a "sampling rule". In a *sequential* design, the nodes are selected one at a time, and for each selection the knowledge from all previous model evaluations is used to identify the point in the parameter domain that promises the most benefit.

The contribution of this chapter lies in the intersection of two disciplines: uncertainty quantification (UQ) and design of computer experiments (DoE).

As pointed out in the introduction, Bayesian parameter inference is a standard UQ problem. Usual approaches are either Markov-Chain Monte-Carlo methods (MCMC) [51, 64] or response surface methods. For the problem setting at hand, MCMC methods require too many function evaluations, which makes them unfeasible. The most common response surface method is stochastic collocation, see Section 3.1.2. Response surface models are widely used for solving Bayesian inverse problems, mostly by directly plugging the response surface into Bayes' theorem [31, 72, 78, 94, 95, 96, 171]. There are also approaches that approximate the posterior distribution itself by a response surface [172].

Some research effort has been made in constructing sequential sampling rules for polynomials in order to sample the model function most densely in areas with high posterior probability [86, 111]. These approaches rely on polynomial response surfaces, which have two disadvantages in this context:

1. Polynomial interpolation is subject to the Runge phenomenon [126]. This means that the interpolant can oscillate in between the nodes if the nodes are not placed with special care. This causes trouble in sequential sampling methods because poorly selected new points can potentially deteriorate a response surface by introducing oscillations. As an example, in the iterative procedure proposed by Li and Marzouk [86], nodes from earlier iterations are ignored. This approach avoids oscillations at the cost of wasting information from previous nodes.
2. Polynomial interpolation methods do not have an explicit error model

for the deviation between surrogate and reference model as a function of the model input. As we will see, such an error model is essential for a sequential sampling method. New nodes can only be selected efficiently if we can pinpoint those areas in the parameter domain that cause the largest errors.

In the field of DoE, there is a number of publications on sequential designs for specific goals, such as exceedance probabilities [6, 10, 12], contour estimation [116, 120], sensitivity indices [141] and optimization [73, 80, 162]. The approaches all have in common that they explicitly consider code uncertainty, see Sections 1.2 and 1.3, by describing the model function with an emulator, see Section 3.2. Recall that I use the term "emulator" according to O'Hagan: an emulator is a random field used as a surrogate [109].

Since the goals are often local features in the parameter domain (e. g. contour estimation would only require function evaluations close to the contour), sequential designs are potentially more efficient than space-filling designs. Sequential designs are based on so-called *figures of merit.* These figures give an estimate of the potential value of knowing the model response in a specific point. Figures of merit have to make a meaningful trade-off between *refining* areas that are known to be important for the goal (e. g. close to the contour) and *exploring* areas that still have a large code uncertainty [73, 130]. The figure of merit used in this work is the *Bayes risk*, which measures the expected loss over the possible shapes of the model function. A precise definition will be given below in Section 6.1.1. The Bayes risk has been successfully applied to other goals, such as exceedance probabilities [10].

The current work is the first application of sequential design of experiments methods on the goal of solving a Bayesian inverse problem, i. e., on calculating a posterior distribution. Emulators have been used in combination with Bayes' theorem [75, 161], but to the best of my knowledge, a sequential design of experiments has not been developed yet.

From Chapter 2, recall that the posterior distribution is proportional to the product of prior and likelihood $\hat{\pi} \propto L(x)\,\pi(x)$ and that the likelihood reads $L(x) = \pi_\varepsilon(z - u(x))$, where π_ε denotes the pdf of the additive error in the error model $Z = u(X) + \varepsilon$. Throughout this chapter, I will add an extra argument to the likelihood function to indicate the model function that has been used in the measurement model: $L(x|u)$. Inserting a random field instead of a function, we obtain $L(x|U)$, which itself is a random field (for each realization of U, we obtain a corresponding likelihood function).

This chapter is structured as follows. In Section 6.1, the sequential design is presented. Then, in Section 6.2, numerical experiments are shown to demonstrate that the sequential design works as expected and achieves a better efficiency than non-sequential methods. Finally, Sections 6.3 and 6.4 provide a discussion and conclusions.

6.1. Sequential Design

A sequential design consists of two components: a strategy and an estimator [10]. A strategy is a rule that selects a new node based on the previous nodes and the corresponding model responses. An estimator is a function that makes an estimate of the target quantity using only model responses at the nodes currently available. The strategy and the estimator have to be adapted to each other and to the overall goal statement.

The estimator and the strategy I present here are designed to minimize the Bayes risk, which is defined as the average loss over a random field that describes the possible shape of the model function. The loss is evaluated via a loss function, which in this work is chosen to be the squared L^2-error in the likelihood estimate.

6.1.1. Loss Function and Bayes Risk

To evaluate the performance of a sequential design, we define a loss function ℓ. It measures the distance between two likelihood functions f and g:

$$\begin{aligned} \ell(f,g) &:= \int_\Omega (f-g)^2(x)\, \mathrm{d}\mu(x) \\ &= \mathrm{E}_X\left[(f(X)-g(X))^2\right]. \end{aligned} \tag{6.1}$$

This is the squared prior-weighted L^2-norm of the difference $f-g$. In the following, this loss function is used to compare likelihood functions. In other words, we do not compare estimators in their ability of estimating the posterior pdf, but in estimating the likelihood function, weighted with the prior. This is justified because a small error in the likelihood can be expected to lead to a small error in the posterior. This approach has two advantages over comparing posterior pdfs directly. First, the mathematical derivations become much simpler. For example, the multiplicative constant in Bayes' theorem is never needed. And second, this approach does not require X to be a continuous random variable. All that is required of X is that one must be able to integrate over the measure μ. If X were a discrete random variable, the approach would be the same, only the integral in Eq. (6.1) would turn into a sum. For future work it would be interesting to compare whether other loss functions perform better. For example one could try the Hellinger distance [149] or KL divergence [79] between estimated posterior and true posterior.

Having the loss function in place, we define the figure of merit as the Bayes risk. Assume that the model function has been evaluated in n nodes and let U_n be the emulator conditioned to these model responses. Let $\tilde{L}_n$ be an estimator for the likelihood that only uses the information from the n model evaluations. If the model function u were known, then the loss $\ell\left(L(\cdot|u), \tilde{L}_n(\cdot)\right)$ would be an error measure for the estimator $\tilde{L}_n$. But since all we know is that u is a realization of U_n, the best we can do is calculate the average loss over the field. This is exactly the definition of the Bayes risk:

$$r_n := \mathrm{E}_{U_n}\left[\ell\left(L(\cdot|U_n), \tilde{L}_n(\cdot)\right)\right]. \tag{6.2}$$

Note that, in this expected value, only the likelihood $L(\cdot|U_n)$ is random, while the likelihood estimator $\tilde{L}_n(\cdot)$ is constant. The Bayes risk can only become small if both the estimator $\tilde{L}_n$ is chosen well and the uncertainty in the field U_n is reduced in the right way (which is done by selecting nodes via the sampling strategy). Therefore, we can use the Bayes risk as an objective function for optimizing both the estimator and the sampling strategy.

In the following, I will first derive the optimal estimator $L_n^\star$, i. e. the estimator that minimizes the Bayes risk in Eq. (6.2) for a fixed set of nodes. Then I will describe a sampling strategy that minimizes the expected Bayes risk in each iteration.

6.1.2. Estimator

With respect to the quadratic loss function [Eq. (6.1)], the best estimator

$$L_n^\star := \underset{\tilde{L}_n}{\operatorname{argmin}}\,[r_n]$$

can be found analytically. Inserting the loss function into Eq. (6.2) and changing the order of integration leads to the expression

$$r_n \quad = \quad \mathrm{E}_X \mathrm{E}_{U_n} \left[\left(L(X|U_n) - \tilde{L}_n(X) \right)^2 \right].$$

The expected value over X becomes minimal if the inner expression is minimized for each $x \in \Omega$ individually. So we obtain

$$L_n^\star(x) = \operatorname*{argmin}_{a \in \mathbb{R}} \mathrm{E}_{U_n} \left[(L(x|U_n) - a)^2 \right].$$

For a fixed x, the likelihood expression $L(x|U_n)$ becomes a random variable. The number minimizing the mean squared deviation from a random variable is the expected value of this variable, so the minimizing estimator is the average likelihood function

$$L_n^\star(x) = \mathrm{E}_{U_n} [L(x|U_n)]. \tag{6.3}$$

With the best estimator, the Bayes risk becomes the variance of $L(x|U_n)$ (with respect to U_n), averaged over the parameter domain Ω with weight $\pi(x)$:

$$\begin{aligned} r_n &= \mathrm{E}_X \mathrm{E}_{U_n} \left[\left(L(x|U_n) - L_n^\star(x) \right)^2 \right] \\ &= \mathrm{E}_X \mathrm{Var}_{U_n} [L(X|U_n)] \end{aligned} \tag{6.4}$$

Note that the best estimator $L_n^\star$ is different from the estimator according to the plug-in approach:

$$L_n^{\text{plug-in}}(x) = L(x|m_n) = L\left(x|\mathrm{E}_{U_n}[U_n]\right). \tag{6.5}$$

The plug-in approach calculates the likelihood of the field average, while the best estimator $L_n^\star$ yields the average of all likelihood functions over the field. The likelihood is a non-linear function, so these two approaches differ strongly.

Alternative Meaning of the Best Estimator

Besides minimizing the average loss, the estimator $L_n^\star$ has a second, very intuitive meaning. We return to the measurement model $Z = u(X) + \varepsilon$

[Eq. (2.6)], but instead of inserting the model function u, we fully acknowledge the uncertainty about u and insert U_n instead. We obtain a new measurement model

$$Z_{\text{new}} = U_n(X) + \varepsilon. \tag{6.6}$$

The deviation between a mean emulator prediction $m_n(x)$ and the measurement value z is now the sum of the measurement error ε and the code uncertainty in U_n: we decompose the random field into its mean $m_n(x) = \mathrm{E}_{U_n}[U_n(x)]$ and deviation $D_n(x) = U_n(x) - m_n(x)$, and for any $x \in \Omega$ define the random variable $\nu(x) := D_n(x) + \varepsilon$. Then, the new measurement model leads to the likelihood function

$$L_{\text{new}}(x) = \pi_{\nu(x)}(z - m_n(x)). \tag{6.7}$$

Applying the law of total probability, we can show that this likelihood coincides with the best estimator $L_n^\star$:

$$\begin{aligned} \pi_{\nu(x)}(z - m_n(x)) &= \mathrm{E}_{D_n(x)}\pi_\varepsilon(z - m_n(x) - D_n(x)) \\ &= \mathrm{E}_{U_n(x)}\pi_\varepsilon(z - U_n(x)) \\ &= \mathrm{E}_{U_n}L(x|U_n) = L_n^\star(x) \end{aligned}$$

So $L_n^\star$ is not only the best estimator for the reference likelihood L, it is also the exact likelihood of a slightly different inference problem, namely the one that explicitly acknowledges the uncertainty due to not knowing the response of u over the whole parameter domain: $L_n^\star = L_{\text{new}}$. That means that if we stop sampling the model function after n evaluations, compute the best estimate likelihood $L_n^\star$ and insert it into Bayes' theorem, $\hat{\pi}_n^\star(x) \propto L_n^\star(x) \cdot \pi(x)$, then the posterior $\hat{\pi}_n^\star$ correctly represents our knowledge about the input parameter. This is an important property of the best estimator because it allows us to stop the calculations after any number of functions evaluations and still provide a meaningful posterior distribution of the input parameters.

If the emulator is Gaussian with mean function m_n and covariance function C_n and if the measurement error is assumed to be normally distributed $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, then we find $\nu(x) \sim \mathcal{N}(0, C_n(x,x) + \sigma_\varepsilon^2)$, which is a convenient way of evaluating the best estimator likelihood via Eq. (6.7) without numerically evaluating an expected value operator, see Eq. (6.3).

To illustrate the difference between the best estimator and the plug-in estimator, a simple example is given in Figure 6.1. The x-axis shows a one-dimensional parameter domain. In the top plot, the unknown model function is shown in black. It is evaluated in the two points marked with circles

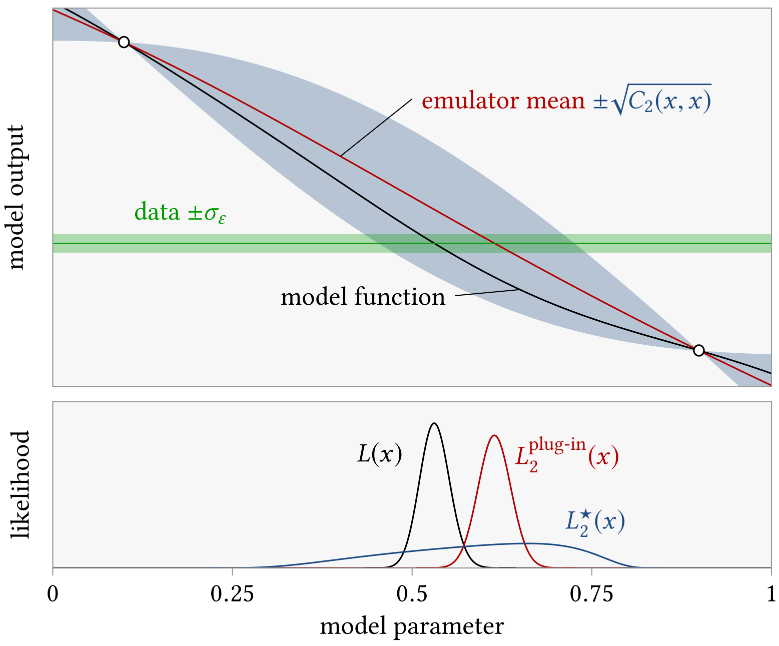

Figure 6.1.: Schematic comparison of the three likelihoods (reference $L(x)$, best estimator $L_2^\star(x)$ and plug-in $L_2^{\text{plug-in}}$).

and the resulting emulator is shown with its mean (red line) and ± one standard deviation (blue shaded area). The bottom plot shows the correct, but unknown likelihood (black), the best-estimate likelihood $L_2^\star$ (blue) and the plug-in likelihood $L_2^{\text{plug-in}}$ (red). Note that the best-estimate likelihood spans the whole region, in which the shaded areas of the emulator and the data intersect, while the plug-in likelihood is biased, being overly confident in the wrong place.

6.1.3. Sampling Strategy

The sampling strategy presented here classifies as a greedy one-step look ahead strategy [10]. In each iteration, the new node is selected as if it were the last one. Each node is selected to minimize the expected Bayes risk under the current knowledge about the model function.

As before, we denote the emulator conditioned to the first n model evaluations by U_n. Let $\hat{x}$ be a candidate point for sampling and assume that $\hat{y}$ is the hypothetical, but uncertain model response at the candidate point. Next, we define the emulator $U_n^{\hat{x},\hat{y}}$ as the emulator U_n conditioned to $U_n(\hat{x}) = \hat{y}$. Then, $\mathrm{E}_X \mathrm{Var}_{U_n^{\hat{x},\hat{y}}} \left[L\left(X | U_n^{\hat{x},\hat{y}} \right) \right]$ is the Bayes risk after the hypothetical model evaluation $U(\hat{x}) = \hat{y}$ (compare to Eq. (6.4)). The model response $\hat{y}$ is, of course, not known beforehand, so we define a random variable describing all possible values of the model response, $\hat{Y} = U_n(\hat{x})$, and define the objective function as the expected Bayes risk over $\hat{Y}$:

$$\hat{r}_n(\hat{x}) := \mathrm{E}_X \mathrm{E}_{\hat{Y}} \mathrm{Var}_{U_n^{\hat{x},\hat{Y}}} \left[L\left(X | U_n^{\hat{x},\hat{Y}} \right) \right]. \tag{6.8}$$

In the following we will refer to $\hat{r}_n(\hat{x})$ as *refinement criterion* because its minimum defines the next node. If the random field U_n is Gaussian and if the measurement error is normally distributed, then parts of the refinement criterion can be evaluated analytically. A brief outline of possible simplifications is given below in Section 6.1.5.

6.1.4. Algorithm

The sequential sampling strategy can be summarized as follows.

1. Define an initial random field U_0 and set counter $n := 0$.
2. Find the new node: $x_{n+1} := \underset{\hat{x} \in \Omega}{\operatorname{argmin}}\, \hat{r}_n(\hat{x})$.
3. Evaluate the model function at the new node: $y_{n+1} := u(x_{n+1})$.
4. Build the new random field U_{n+1} by conditioning U_n to (x_{n+1}, y_{n+1}).
5. Increase the counter n by 1.
6. Jump back to step 2 until an exit condition is met.
7. Output $L_n^\star$.

Possible exit conditions are a maximum number of function evaluations or a threshold for the Bayes risk r_n.

6.1.5. Simplifications for Gaussian Process Emulators

The main difficulty in applying the presented sequential sampling strategy lies in the calculation of the refinement criterion $\hat{r}_n$ [Eq. (6.8)] in order to find its minimum. In this section, I briefly outline two possibly ways of simplifying the calculation of the refinement criterion. These simplifications can be made if the random field is Gaussian and the measurement error is normally distributed.

First, the two inner integral operators $\mathrm{E}_{\hat{Y}}\mathrm{Var}_{U_n^{\hat{x},\hat{Y}}}[\ldots]$ in Eq. (6.8) can be evaluated analytically by using the following three properties: (1) the pdf of the measurement error ε is a Gaussian curve, (2) at any point x, the emulator response $U_k^{\hat{x},\hat{Y}}(x)$ is normally distributed and (3) the relationship between a possible model response $\hat{Y}$ and the mean of the emulator response at a different point $U_n^{\hat{x},\hat{Y}}(x)$ is linear, see the Kriging system [Eq. (3.19)]. These three properties turn the two inner integrals into integrals over the product of Gaussian curves, which can be calculated analytically.

Second, it is possible to reformulate the minimization problem to an equivalent maximization problem and eliminate one of the integrals on the way: by applying the law of total variance we find

$$\begin{aligned}\mathrm{Var}_{U_n}\left[L\left(x|U_n\right)\right] &= \mathrm{E}_{\hat{Y}}\mathrm{Var}_{U_n^{\hat{x},\hat{Y}}}\left[L\left(x|U_n^{\hat{x},\hat{Y}}\right)\right] \\ &+ \mathrm{Var}_{\hat{Y}}\mathrm{E}_{U_n^{\hat{x},\hat{Y}}}\left[L\left(x|U_n^{\hat{x},\hat{Y}}\right)\right]. \qquad (6.9)\end{aligned}$$

The left hand side term is independent of $\hat{x}$, so the two terms on the right hand side (as functions of $\hat{x}$) add up to a constant. The first term on the right hand side is the inner part of the refinement criterion $\hat{r}_n(\hat{x})$. If we replace it by the second term on the right hand side, we obtain a function whose maximum is at the same point as the minimum of the refinement criterion. This replacement is possible also for non-Gaussian emulators. If the emulator is Gaussian, then additionally the inner expected value with respect to $U_n^{\hat{x},\hat{Y}}$ can be eliminated: it has the same form as the best estimator [Eq. (6.3)] and can therefore be calculated using an expression analogous to Eq. (6.7). After this step, only the variance operator remains, which in the Gaussian case can be calculated analytically.

To summarize: in the Gaussian case, only the integral operator over X remains and needs to be evaluated numerically. In the general, non-Gaussian case, we can at least eliminate one of the three integral operators if we are able to find the pdf of the deviation $v\left(x\right)$, see Eq. (6.7)

6.2. Numerical Experiments

In this section, I demonstrate the sequential design in a couple of test cases. The first two test cases use Gaussian process emulators, while the third test case uses a non-Gaussian emulator that includes a discontinuity. The gpe used in the first two examples are of a certain kind. They are the mix of two random fields:

$$U_0(x) = \alpha P(x) + (1 - \alpha) S(x), \tag{6.10}$$

with $\alpha \in [0, 1]$.

The first part, P, is a truncated polynomial chaos expansion with random coefficients (see Section 3.1.2)

$$P(x) = \sum_{i=1}^{p} \lambda_i X_i \Psi_i(x).$$

Here, the functions Ψ_i form an orthonormal basis of polynomials. The coefficients λ_i form a decaying spectrum and the X_i are i.i.d. random variables $X_i \sim \mathcal{N}(0, 1)$. In the following examples, I chose the basis of polynomials to form a total degree space of degree g. Furthermore, the spectrum is chosen, such that λ_i is a function only of the degree of Ψ_i, so it can be fully specified as a $(g + 1)$-sized vector. The decay rate of the spectrum reflects a smoothness assumption of the model function u.

The second part, S, is a zero-mean stationary field with a Gaussian-type correlation function: $m_S = 0$ and $C_S(x, x') = \exp\left(-\frac{1}{2}\|x - x'\|^2/l\right)$, where l is the correlation length.

The two parts P and S are assumed to be independent so the field mean and covariance of the two parts add up. To fully describe a field of this type, one needs to specify α, g, λ and l.

6.2.1. Synthetic One-Dimensional Gaussian Random Field

In this example, we examine a problem with one input parameter and one model output quantity. To isolate the performance of the sequential design from possible modeling issues, the random field U is defined first and then the model function u is defined by drawing one realization from U. In practice, of course, the process works in reverse: a model function is given and the modeler has to specify a random field from which this model function could be a realization.

We define the input parameter domain as $\Omega = [-1, 1]$ and let the parameter be uniformly distributed on Ω. The random field is modeled as given in Eq. (6.10), with the parameters $\alpha = 0.99$, $g = 50$, $\lambda_i = \mathrm{e}^{-i}$ and $l = 0.1$. The measurement data value z is generated by drawing a sample from the input distribution and inserting it into the measurement model $Z = u(X) + \varepsilon$ [Eq. (2.6)]. The measurement error ε is normally distributed with zero mean and variance $\sigma_\varepsilon^2 = 0.001$. For calculating $\hat{r}_n$ and for finding its minimum, the domain Ω is discretized by 501 equispaced points.

A plot of the iterative sampling procedure is shown in Figure 6.2. Each group of plots shows the current emulator mean and standard deviation together with the data and the nodes (left), the current estimated posterior in comparison to the reference posterior (top right) and the refinement criterion $\hat{r}_n$ (bottom right). The figure shows the results after 1, 3, 6 and 12 model evaluations.

The first model evaluation is done exactly in the center of the domain. After that step, the refinement criterion is symmetric and the point on the right is selected by random choice. By coincidence, this is also the area where the model crosses the data. Next, the third node is spent on refining this area. The following three nodes (4, 5 and 6) lie in the area of the second mode of the posterior. After that, also the parts further away are explored. In the last plot, there is still considerable uncertainty in the emulator, but only in those parts that are far away from the data and therefore do not have a large impact on the posterior.

Note that the refinement criterion is discontinuous in the points that have already been evaluated. Evaluating in a single node twice has no benefit and therefore does not decrease the expected Bayes risk. Evaluating in two different nodes that are very closely together, however, yields a gradient information, which is useful information no matter how close the two nodes are to each other.

We now compare the performance of the sequential design with two non-sequential methods: first, an emulator conditioned on equidistant nodes and, second, a polynomial response surface. In this comparison, all methods use 12 model evaluations. For the emulator, we use the same prior field as the iterative method above, but we use 12 equidistant nodes instead of

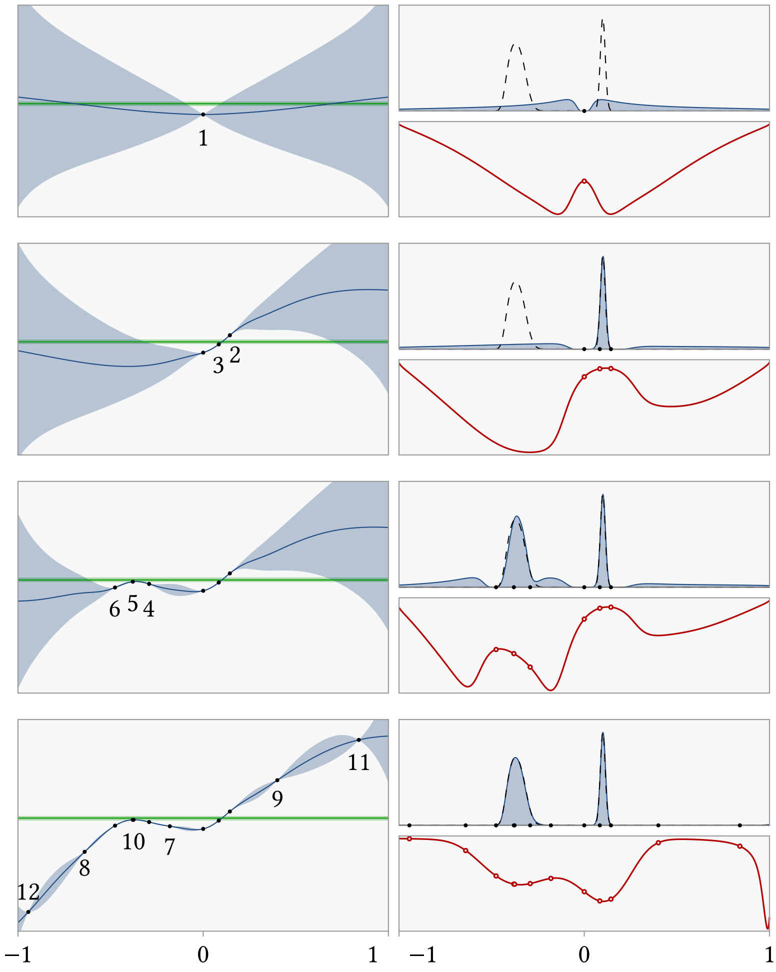

Figure 6.2.: Iteration procedure for synthetic 1d-example, results after 1, 3, 6 and 12 model evaluations. All plots: x-Axis is the parameter domain $[-1, 1]$. Left: emulator ± one estimation standard deviation (blue) and measurement data ± measurement standard deviation (green). Right top: estimated posterior (blue shaded) and true posterior (dashed line). Right bottom: refinement criterion $\hat{r}_n$ (vertically normalized to fit into the box).

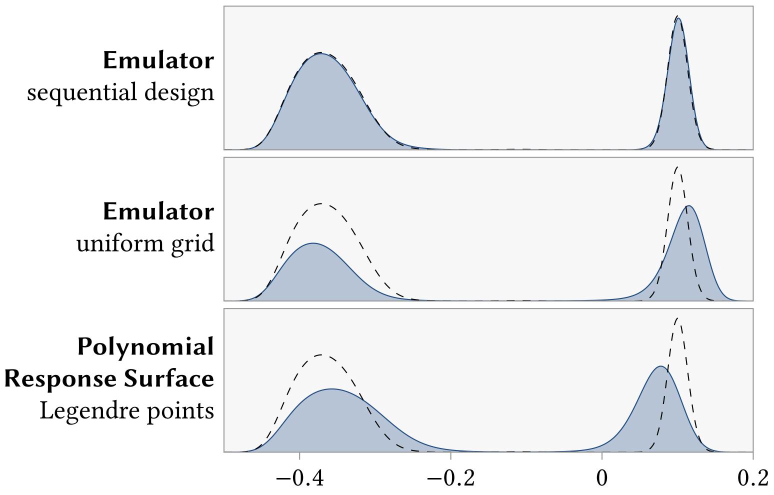

Figure 6.3.: Posteriors calculated with (i) the presented sequential design, (ii) the emulator with equidistant nodes and (iii) a polynomial response surface. The true posterior is shown as a dashed line for comparison. The plots are clipped to show only the relevant parts of the parameter domain Ω.

sequentially chosen ones. Then, the best estimate posterior is calculated. For the polynomial, we use the nodes of a 12-point Gaussian quadrature rule, in this case the Legendre points, and interpolate the model response with a polynomial of degree 11. The polynomial is compared in terms of its plug-in posterior. Figure 6.3 shows the posterior distributions obtained with the different methods. The sequential design clearly performs best. Both other methods produce errors in the shape of both peaks.

In summary, the sequential sampling strategy works well in this test problem, being more efficient in calculating the posterior distribution than the two non-sequential methods.

6.2.2. Heat Equation

As a second example, we examine a simulation-based problem with a two-dimensional parameter space and an 18-dimensional output. This is the same problem as examined by Li and Marzouk [86]. Recall that the spatial coordinate is denoted by ξ (here two-dimensional) and time is denoted by τ. We consider the two-dimensional heat equation on the unit square $[0,1]^2$:

$$\frac{\partial\theta}{\partial\tau} = \nabla^2\theta(\xi,\tau) + s(\xi),$$

where θ is temperature, s is a heat source term and ∇^2 is the Laplace operator. The source term is given as

$$s(\xi) = \frac{s_0}{2\pi h^2}\exp\left(\frac{-|X-\xi|^2}{2h^2}\right),$$

with an uncertain heat source location X. We select a uniform prior for the source location: $X \sim \mathcal{U}([0,1]^2)$. The parameter domain of X and the spatial domain of ξ are the same in this example. The other parameters are deterministic with $s_0 = 2$ and $h = 0.05$ and the initial condition is set to $\theta(\xi, 0) = 0$. On the boundary, we impose no-flux boundary conditions.

The model function u consists of the temperature θ at two time steps $\tau_1 = 0.1$ and $\tau_2 = 0.2$, evaluated at 9 points in the domain, resulting in a total of 18 output values. The 9 positions are $\{0, 0.5, 1\}^2$. For each measurement, $i = 1, \dots, 18$, an additive and independent normally distributed error $\varepsilon_i \sim \mathcal{N}(0, 0.1^2)$ is assumed leading to the measurement model $Z_i = u_i(X) + \varepsilon_i$. A vector of virtual measurement data is generated by setting the source position $X = (0.25, 0.25)$, solving the model equation and adding random noise according to the error model. In the forward model, the heat equation was solved on a spatial 512×512 grid and, for time integration, the solver used the Euler forward method for the source term and an analytic solution on the Fourier domain for the diffusion term.

The reference posterior is calculated on a 101×101 grid. Note that the computational domain of the heat equation and the parameter domain of the source location are equal by coincidence only. In principle, they are independent and they are therefore discretized with different resolutions. The reference posterior is unimodal with its peak close to the true source position $(0.25, 0.25)$.

We now apply the sequential design method to this problem. The initial random field is again a mix of a stationary field and a polynomial with uncertain coefficients. The exact parameters are given in Table 6.1 in the row called *hybrid*. The optimization problem of finding the minimal $\hat{r}_n$ was

Name	l	g	λ	α
hybrid	0.2	10	$(10, e^{-1}, e^{-2}, \dots, e^{-10})$	0.95
polynomial	–	10	$(10, e^{-1}, e^{-2}, \dots, e^{-10})$	1
small-l	0.1	–	–	0
large-l	0.3	–	–	0
uncertain mean	0.1	0	(10)	0.5
linear trend	0.1	1	$(10, e^{-1})$	0.5

Table 6.1.: Parameters of the various random fields.

solved by complete enumeration on a 101×101 grid. Figure 6.4 shows the sequence of nodes for this field together with the location of the posterior peak indicated by contour lines. Note that the diagrams only show the subdomain $[0, 0.75]^2$ because all nodes are contained in this area. The sampling starts in the center of the domain and from there moves towards the posterior peak within the first six evaluations. From then on, many nodes are spent on refining around the posterior peak. Only later after about 20 evaluations, some nodes are spent on exploring the more remote parts of the parameter domain. As we will see below, this is also the moment that we obtain a new increase in accuracy.

Contour lines of the reference posterior and of the estimated posterior after 40 evaluations are shown in Figure 6.5. The only visible deviation lies in the right part of the plot. By close inspection of Figure 6.4, we can see that this is also a part where only few function evaluations are placed.

Sensitivity to the Choice of Random Field

In this section, we investigate the sensitivity of the result to the choice of random field. We compare six different random fields. The individual parameters are given in Table 6.1. The case *hybrid* is the one already used above. It is a mix of a polynomial with a stationary part that has a medium correlation length. Then, case *polynomial* is the same field, but without the stationary part. This results in a random field whose realizations are polynomials. Kriging with this field is not to be confused with polynomial interpolation. The two cases *small-l* and *large-l* are purely stationary fields with different correlation lengths. Finally, the two fields *uncertain mean* and *linear trend* are the same as *small-l* but include additional constant and linear trend parts.

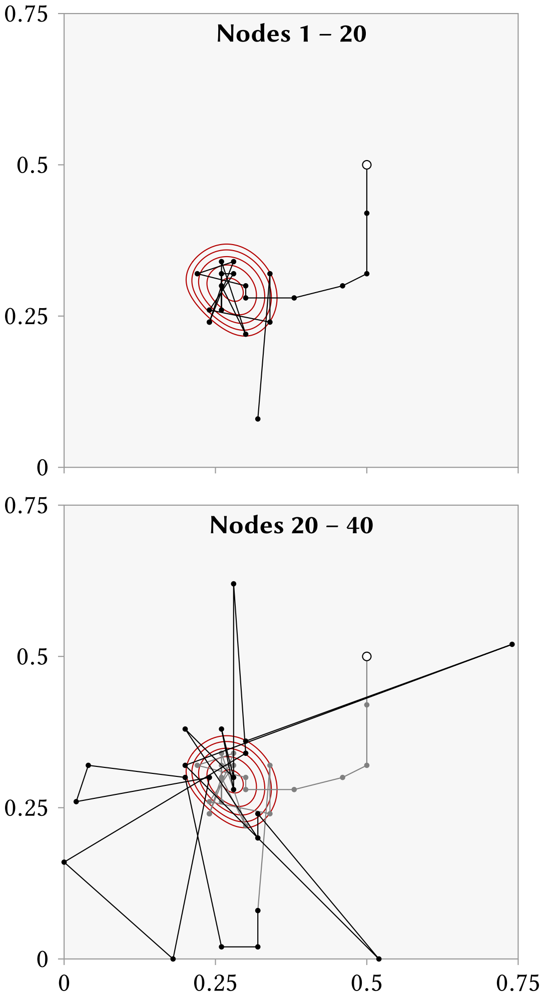

Figure 6.4.: Sequence of nodes for random field of type *hybrid*. Top: nodes 1 to 20. Bottom: nodes 20 to 40 (and nodes 1 to 20 in gray). The peak of the posterior is indicated by red contour lines. Both diagrams are restricted to the subdomain $[0, 0.75]^2$ because all nodes lie within this area.

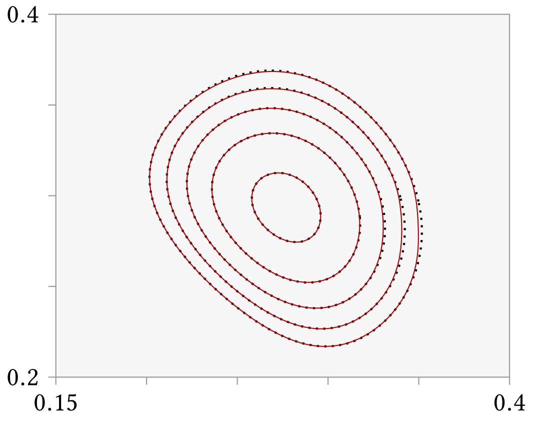

Figure 6.5.: Comparison of the true posterior (black, dotted contours) and the estimated posterior after 40 model evaluations (red, solid contours).

The error behavior is shown in Figure 6.6. The upper plot shows the error in terms of the loss function between reference likelihood and estimated likelihood $\ell\left(L\left(\cdot|u\right), L_n^\star\left(\cdot\right)\right)$ and the bottom plot shows the error in terms of the KL divergence between reference posterior and estimated posterior, see Section 2.2.3. Recall that, ultimately, we are interested in achieving small errors in the KL divergence, but the sequential design is constructed to minimize the loss function instead. We can see that overall, these two goals align well and, in both plots, the individual data lines are roughly in the same order. The main difference is that the loss function converges monotonically, while the KL divergence in some cases shows an increasing error first before the error finally decreases.

It turns out that the three fields that assume a higher smoothness converge fastest, namely the cases *hybrid*, *polynomial* and *large-l*. However, due to the smoothness assumption, Kriging becomes increasingly instable with an increasing number of nodes. Therefore, the data line for the case *polynomial* and *large-l* stop early. If the correlation length is increased further, then the method breaks down even earlier. Gaussian-type correlation functions are known for producing ill-conditioned Kriging systems [1].

After a stagnation phase, the hybrid field and the polynomial field have

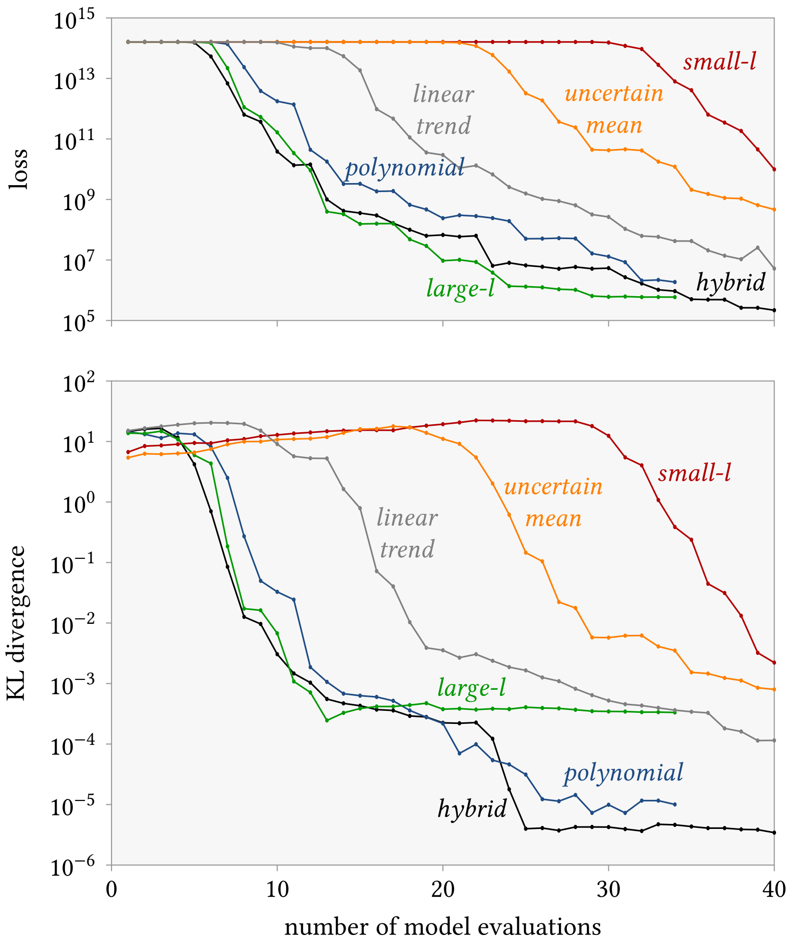

Figure 6.6.: Error behavior of different random fields in comparison. Upper plot: errors in terms of the loss function. Lower plot: errors in terms of the KL divergence to the reference posterior.

a second phase of improved accuracy, which happens at about 24 nodes. This is roughly the iteration in which the sampling method stops refining around the peak and starts exploring the rest of the parameter domain.

The fields with smaller correlation lengths are a bit less efficient, requiring a larger number of evaluations to reach a comparable accuracy. Interestingly, all off the error plots drop from an error of 10^1 to 10^{-3} within relatively few function evaluations and have a plateau before and after this drop. This is the phase during the iteration, in which the sampling method has found the posterior peak and evaluations are spent on refining this peak.

It is interesting to compare the cases *small-l*, *uncertain mean* and *linear trend*. These all have the same stationary part in the field, but different polynomial parts: the first one has no polynomial part, the second one has a large constant and the third one a constant and a linear part. As seen in the error plot, these polynomial parts improve the efficiency. With the polynomial parts, the method is able to find the posterior peak earlier. This is because the additional polynomial parts allow the method to extrapolate to some extent, while purely stationary random fields can only make statements close to the nodes. As a result, the field *small-l* takes the smallest steps in the parameter domain and therefore reaches the posterior peak later than the other fields.

All of the error plots stagnate at some point, the best one reaching an error of 10^{-5}. At this point, the numerical stability of the Kriging procedure hinders the further improvement of accuracy.

In a test that is not further reported in the error plot, I tested the performance of a random field with an exponential correlation function: $C(x, y) = \exp\left[-\left|x - y\right| / l\right]$. This field lead to very bad results because of the lack of a smoothness assumption. While the heat equation is expected to behave smoothly with respect to the position of the source term, the realizations of a random field with an exponential covariance function are continuous but not differentiable. Therefore and without surprise, such a random field is not an appropriate description of the model function.

But even if it were appropriate, and the model function were not differentiable, then point evaluations would provide very little information about the actual shape of the model function. Therefore, a very large number of model evaluations would be required. In the case of a non-differentiable model function, other surrogate methods such as polynomial chaos expansions would struggle as well because the expansion coefficients would decay only slowly.

This comparison of different random fields demonstrates that the performance of the sequential sampling strategy is sensitive to the choice of random field. For demonstration purposes, the different random fields were picked by hand and compared. In practice, of course, the choice of the random field should be done in a less subjective manner. Possible approaches for doing so are given below in the discussion in Section 6.3.2.

Comparison to Other Sampling Methods

Next, we compare the performance of the sequential design with the two non-adaptive methods we already used in the first test case (gpe and polynomial response surface). For the polynomial, we sample the model function on a grid constructed from the zeros of Legendre polynomials. The numbers of nodes are therefore $2^2, 3^2, \ldots$. For the gpe, we use a greedy sampling strategy, each step minimizing overall code uncertainty, which in terms of the estimation variance is $\int_\Omega C_n(x,x)\,\pi(x)\,\mathrm{d}x$, using the conditional covariance function C_n, see Eq. (3.19). The conditional covariance C_n is independent of the actual model response (see Eq. (3.19) in Section 3.2), so this sampling strategy is effectively non-adaptive.

Figure 6.7 shows the error behavior in terms of the KL divergence. The data point with 9 nodes for the polynomial failed to calculate a finite KL divergence, possibly because the corresponding likelihood was numerically rounded to zero somewhere in the domain. Therefore, this point is omitted from the plot.

In the investigated range of nodes, both methods yield significantly worse results compared to the sequential design method.

6.2.3. Jump-Detection Problem

Finally, we examine an example where a non-Gaussian field is a suitable emulator. For demonstration purposes, I chose the example such that all integrals in the refinement criterion $\hat{r}_n$ can be calculated analytically. The model function u is known to have the shape of a Heaviside function, yet with unknown jump location. Let $\Omega = [0,1]$ be the parameter domain and $T \sim \mathcal{U}(\Omega)$ be a random point in this domain. We define the random field as

$$U_0(x) = \begin{cases} 0 & \text{if } x < T \\ 1 & \text{if } x \geq T. \end{cases}$$

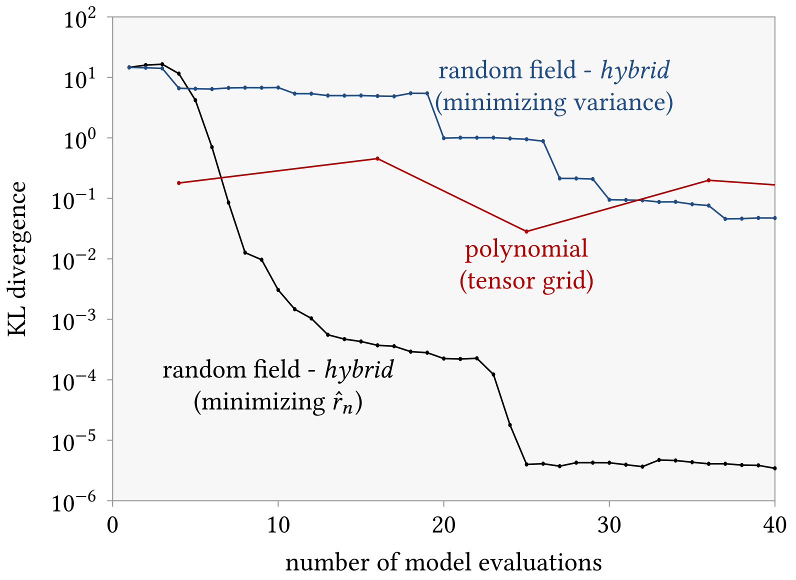

Figure 6.7.: Error behavior of different sampling methods in comparison.

This field is non-Gaussian. We assume that the input parameter X is uniformly distributed on Ω and the measurement model is $Z = u(X)$, i. e., without measurement error. Note that X is not the jump location, but the parameter inserted into the model function. But since the random field is effectively one-dimensional, being parametrized by the jump location T, knowing T is equivalent to knowing the posterior of X. As we will see below, the resulting sequential design is independent of the value of the measurement Z.

The functional form of criterion $\hat{r}_n$ can be derived analytically. The derivation is given in Appendix B and here I only present the result. Assume that, at some point in the iteration, the model has been evaluated at nodes $x_1, \ldots, x_n$. We define the two existing nodes closest to the jump as

$$\begin{aligned} a &= \max\{x_i | u(x_i) = 0\} \\ b &= \min\{x_i | u(x_i) = 1\}. \end{aligned}$$

These two act as bounds for the value of T because $T \in [a, b]$. Next we

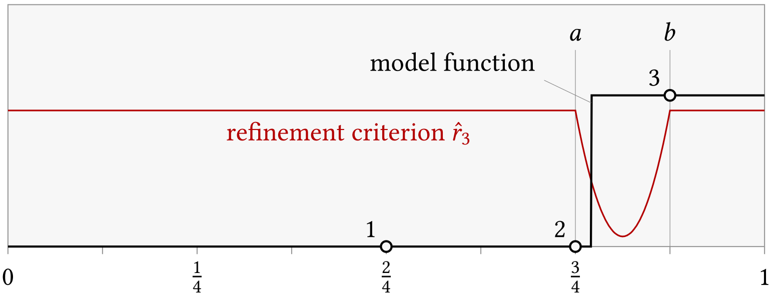

Figure 6.8.: Third step of the iterative sampling process. Model function (black) with nodes (circles) and the refinement criterion for the next step (red). The points a and b are the two nodes closest to the jump.

define a normalization function h_{ab}

$$h_{ab}(x) := \begin{cases} 0 & \text{if } x < a \\ \frac{x-a}{b-a} & \text{if } x \in [a, b] \\ 1 & \text{if } x > b. \end{cases}$$

Then the criterion $\hat{r}_n$ has the analytical form

$$\hat{r}_n(\hat{x}) = C - \frac{1}{3}(b-a) \cdot h_{ab}(\hat{x})(1 - h_{ab}(\hat{x})),$$

where C is a constant that is independent from $\hat{x}$. Surprisingly, the criterion is independent of the actual value of Z.

Figure 6.8 shows the third step in the iterative refinement procedure, where the jump location was set to $t = 0.77$. The model function is shown in black. It is evaluated in the three nodes shown as circles. The red line shows the refinement criterion $\hat{r}_3$ for the next iteration. The refinement criterion is constant outside the interval $[a, b]$ and a symmetric parabola inside the interval. Therefore, the next node is placed in the center $(a + b)/2$. This is true for every iteration, so the sequential sampling procedure is equivalent to the bisection method.

This example shows that the method works in an intuitively plausible way, even for non-Gaussian random fields. Furthermore, the specific example of jump detection indicates that our method could also effectively work for model functions with discontinuities.

Functions with discontinuities typically cause difficulties for UQ methods because they cannot efficiently be approximated with polynomials, and if the position of the discontinuity is unknown, a Gaussian process emulator is not an appropriate representation for such a function, either.

While in the given example, the discontinuity could simply be described by one parameter, a parametrization of the possible shape of a discontinuity becomes more complex if the parameter domain is higher dimensional. This complicates both the choice of a suitable random field and the practical calculation of the integrals involved in the refinement criterion. In summary, the idea of describing discontinuous functions with non-Gaussian fields is conceptually elegant, but more research is needed to overcome the technical difficulties and to make this approach practical.

6.3. Discussion

In this section, we discuss limitations and possible extensions of the presented sequential design.

6.3.1. Limitations

The presented sequential design has two numerical difficulties.

First, calculating the refinement criterion $\hat{r}_n$ involves three integrals. If the random field is Gaussian, then two of them can be evaluated analytically. But in the general non-Gaussian case, the calculation of $\hat{r}_n$ would be tedious and would possibly involve sampling of the random field. In practice, one might face situations, where the use of a non-Gaussian field is desired. For example, if the correlation length of the model function is not known, then one could include the correlation length as an uncertain meta-parameter (instead of selecting the maximum likelihood value). For a given correlation length, such field would be conditionally Gaussian, but, as a whole, it would be non-Gaussian. In geostatistics, the approach of including uncertain meta-parameters is called Bayesian geostatistics [41, 76, 92, 106]. Another example of a naturally occurring non-Gaussian random field would be a model function with a jump discontinuity at an uncertain position as presented in the third numerical example.

Second, for very large numbers of nodes, the Kriging system in Eq. (3.19) becomes ill-conditioned. The second numerical experiment above is an example for this problem. With a straightforward implementation of the Kriging system, the two very smooth fields in Figure 6.6 could not be conditioned to more than 33 nodes. This is purely a numerical problem caused by the high smoothness in the assumed covariance function [1]. Analytically, the Kriging equations have a unique solution, as long as the random field has positive variance. In the field of geostatistics, a valid approach to improving the condition of the Kriging system is to introduce a measurement error [40]. In the case of a computer experiment, however, the model output is deterministic, so a measurement error cannot be justified. I have two more ideas for improving the condition of the Kriging system, but have not implemented or tested these: First, it might be possible to solve the Kriging system more stably using an iterative Levenberg-Marquard type solver as proposed in a similar fashion by Nowak and Cirpka [105]. Second, it might be helpful to add nodes to the design that are more evenly spread over the parameter domain. These nodes would not be optimal with respect to the goal of solving an inverse problem, but they might help improve the condition of the Kriging system.

The possible bad condition in Kriging is qualitatively different from the Runge phenomenon for polynomials. According to the Runge phenomenon, polynomials that interpolate certain points exhibit strong oscillations between the points [126]. The problem with the Kriging system is that the actual interpolant is difficult to calculate. Oscillations between the nodes are not expected.

6.3.2. Possible Extensions

The loss function introduced in Section 6.1.1 was only applied to the likelihood term, not to the full posterior distribution. While this simplifies many calculations, one could argue that a proper loss function should rather compare posteriors, not likelihoods. One possible problem could be seen in the second numerical experiment. In Figure 6.6, the loss decreases monotonically, while the KL divergence shows a slow increase in error at early times. For future research it would be interesting to investigate, whether it is possible to use the KL divergence of the posterior distribution directly as the loss function and whether this is enough to avoid a non-monotonic error behavior.

Furthermore, the second numerical example showed that the sequential design is sensitive to the choice of random field. As an aid for the modeler I suggest two possible approaches to make the choice of random field less subjective. The first approach is to add an initial sampling phase before the sequential design. In this phase, the model function is evaluated on a space-filling set of points. The field parameters (correlation length, smoothness parameters,...) of the a priori field are then estimated using these evaluations, e. g. via the maximum likelihood method [93]. Furthermore, Jones et al. recommend to validate the resulting random field [73]. The second possible approach is to model the field parameters as random variables themselves. The resulting random field would be non-Gaussian in general. The identification of the field parameters is not done in a separate phase, but during the sequential sampling. In the sequential design, the knowledge about the field parameters would be updated automatically and only to that extent that is important to the final goal, the solution of the Bayesian inverse problem. In geostatistics, this approach is known as Bayesian geostatistical design [41, 92, 106]. This approach, of course, has the downside that non-Gaussian random fields have to be handled.

6.4. Conclusions

In this chapter, I presented a sequential design of computer experiments for solving Bayesian inverse problems. The design explicitly accounts for code uncertainty by describing the model function as an emulator. The sampling strategy classifies as a greedy one-step look ahead method and selects the nodes to minimize the expected Bayes risk based on a quadratic loss function. The likelihood estimator fully acknowledges code uncertainty. Therefore, at any iteration, the sequential sampling can be stopped and the result will be a meaningful posterior distribution.

The design was tested in three numerical examples. The first two examples demonstrate that the presented method is more efficient than space-filling designs. Furthermore, the second example shows that the performance of the design is sensitive to the choice of emulator. This suggests that reliable methods for choosing emulator parameters are required. Finally, the third example showed that the proposed method even works for non-Gaussian random fields. This last example also indicates that the method can potentially handle model functions with discontinuities.

7. The Use of Low-Fidelity Models in Bayesian Updating

In this chapter, we have a closer look at the use of low-fidelity models for the task of Bayesian updating.

As stated in the introduction in Section 1.1, we assume that for a given simulation problem, we have a model that describes the physical processes correctly: a model without model error. Within this chapter, we call this model the *reference model*. Furthermore, we assume that a low-fidelity model for the same problem is available, i. e., a reduced version of the reference model. Such a reduced model can be obtained by either coarsening the time- or space-discretization or by simplifying the physical representation of the underlying problem, i. e., by making simplifying assumptions or by neglecting less important physical processes. Due to these simplifications, a reduced model is typically much faster than the reference model, but introduces a model error. This is particularly interesting in stochastic applications, where the model function is evaluated repeatedly, such as a Monte-Carlo simulation. Under time constraints, a reduced model allows a larger number of model evaluations, resulting in a smaller stochastic error (e. g., Monte-Carlo error, see Section 2.1.4). In some cases, the simplified form of the reduced model even allows the derivation of a close-form expression for the uncertainty propagation problem, eliminating the stochastic error completely [144, 150, 154]. The balance between model error and stochastic error leads to the following model-selection problem: which model achieves the smaller total error (=model error plus stochastic error)? Leube et al. discussed this question under the term *optimal allocation of computer resources* [83]. It is basically an optimization problem over a set of two options.

This is a very specific view on model selection and it differs from much of the existing literature on model selection and model averaging. Let me briefly outline the differences. The field of model selection is well established [17, 19, 26]; it it applied in various areas such as psychology [173], hydrology [52], ecology [3] and sociology [119]. The main focus of such studies is to identify a model with highest accuracy or best predictive power, without considering computational costs. In this chapter, we pose a different question: given constraints in computing time, what model should be used? This is a question of *efficiency* rather than accuracy. The efficiency aspect of model selection also lies outside the scope of Bayesian model averaging (BMA) [45, 65, 102], and multilevel Monte Carlo (MLMC) simulations [27, 59]. BMA assigns a weight to each model, which is equal to its probability of predicting given data, without considering the cost-accuracy tradeoff. In our analysis the fidelity of individual models is known a priori. MLMC uses multiple models in a hierarchical way to increase the efficiency of stochastic calculations. Throughout this chapter, I will use the terms *model selection* and *model-selection problem* only with respect to the efficiency aspect explained above.

When measurement data are available, Bayesian updating adds another aspect to the model-selection problem. This is best explained with Figure 7.1, which shows the general workflow used within this chapter. We start with the input parameters X, which are the same for both the reference model u and the reduced model $\tilde{u}$. Both models yield an output, Y and $\tilde{Y}$, respectively. From the output we can derive two further quantities: the measurement data, which is a function of the output and random noise, $Z = f_1(Y, \varepsilon)$; and a quantity of interest (QoI), which is a function of the output, $\text{QoI} = f_2(Y)$. In the updating step, we assimilate measurement data to update the knowledge about the QoI.

We do not attempt to update the knowledge about the input parameters X (which would be an inverse problem). Knowing that the reduced model $\tilde{u}$ contains a model error, we would only obtain effective parameters, and not the true posterior distribution of X. Since we only consider the relationship between the output, the measurement and the QoI and use the same

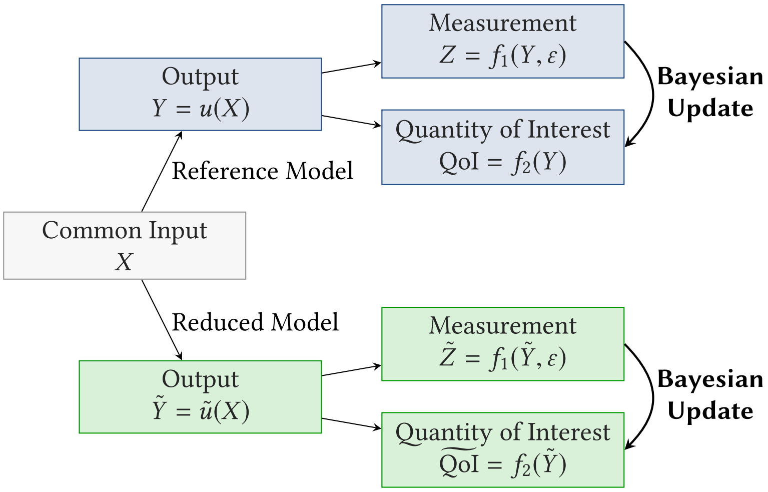

Figure 7.1.: Flow of information for the numerical experiments in this chapter.

functions f_1 and f_2, the functional relation between the measurement data and the QoI is not affected by the model error of $\tilde{u}$.

In this setup, the distributions of Y and $\tilde{Y}$ can be regarded the prior distributions. In that respect, we obtain two different Bayesian updating problems: they have the same measurement model and the same observations, but two different priors.

Coming back to the model-selection problem, we can expect the amount of measurement data to have an influence on both the model error and the stochastic error:

If more and more data become available, the likelihood of the measurement becomes more and more narrow, so that the significance of the prior is expected to diminish. This means that the influence of the model selection becomes smaller and therefore the effect of the model error decreases.

Moreover, the calculation of a posterior distribution becomes increasingly sampling intensive as the amount of available data increases. As more data become available, the percentage of realizations that match the data decreases and therefore larger sample sizes are required. Or, if the sample size is kept constant, the stochastic error increases.

In summary, we can expect Bayesian updating to decrease the model error of a reduced-complexity model and to increase the stochastic error of all calculations. These two effects change the balance between the model errors and stochastic errors and, therefore, affect the optimal model selection. The aim of this chapter is to understand the effect data has on the optimal model selection. Furthermore, we will investigate whether we can derive practical model selection guidelines from these observations.

This chapter is structured around one numerical example. I will first describe the numerical setup in detail and present its results. Only afterwards, a more general approach is taken to the model-selection problem.

7.1. Simulation Setup

As the exemplary model-selection problem, we consider an infiltration process. The geometrical setup is shown in Figure 7.2. We consider a two-dimensional domain $\mathbb{D} = [-1, 1] \times [0 \times \infty]$. The two spatial coordinates are denoted by ξ (horizontal) and ζ (vertical, positive axis pointing downwards). In the initial state, the domain is filled with dry soil. The infiltration is driven by a constant pressure head ψ_0 of ponding water at the soil's surface ($\zeta = 0$). Uncertainty lies in the soil properties. The soil is assumed to be inhomogeneous, so the material properties vary over ξ and ζ. The QoI is the total amount of infiltrated water after 30 min of infiltration.

To decrease the uncertainty in the prediction of the QoI, measurement data from soil moisture sensors are collected and used for Bayesian updating. This is a general Bayesian updating problem as presented in Figure 1.2 in Section 1.1. The measurements are taken at time $\tau = 30$ min and the sensors are located 10 cm below the soil surface. The number of sensors varies over the different experiments.

Such an infiltration process can be described by two alternative models: the Richards equation, which we regard as the reference model, and the Green-Ampt model, which is a reduced version of the reference model. In the following two sections, I provide details about these two models. After that, I briefly comment on the numerical implementation of both models, as well as the updating procedure.

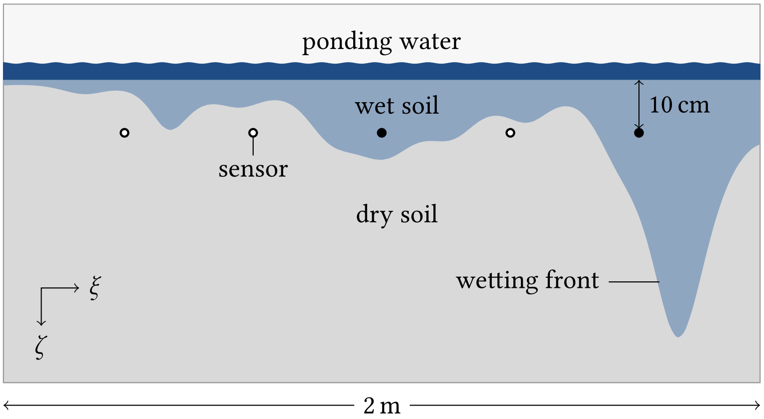

Figure 7.2.: Problem setup.

7.1.1. Richards Equation

At any point $(\xi, \zeta)^\top$ in the flow domain $\mathbb{D}$, the temporal evolution of water content $\theta(\xi, \zeta, \tau) : \mathbb{D}\times\mathbb{R}^+ \longrightarrow [\theta_\mathrm{i}, \phi]$ and pressure head $\psi(\xi, \zeta, \tau) : \mathbb{D}\times\mathbb{R}^+ \longrightarrow \mathbb{R}$ are described by the Richards equation [155]

$$\frac{\partial \theta}{\partial \tau} = \nabla \cdot (K \nabla \psi) - \frac{\partial K}{\partial \zeta}, \tag{7.1}$$

where $K(\xi, \zeta, \theta)$ is the soil hydraulic conductivity. The water content θ lies in the interval between the irreducible water content θ_i and the porosity ϕ. The Richards equation is supplemented by two constitutive relations $K = K_\mathrm{s}(\xi, \zeta) K_\mathrm{r}(\xi, \zeta, \psi)$ and $\theta = f(\psi)$, where K_s and K_r are the saturated and relative hydraulic conductivities, respectively. We employ the van Genuchten constitutive model [155],

$$\begin{aligned} K_\mathrm{r} &= \frac{\left[1 - \psi_\mathrm{d}^{mn}\left(1+\psi_\mathrm{d}^{n}\right)^{-m}\right]^2}{\left(1+\psi_\mathrm{d}^{n}\right)^{m/2}}, \\ \frac{\theta-\theta_\mathrm{i}}{\phi-\theta_\mathrm{i}} &= \frac{1}{(1+\psi_\mathrm{d}^{n})^{m}}, \\ \psi_\mathrm{d} &= \alpha\,|\psi|\,, \\ m &= 1-\frac{1}{n}. \end{aligned} \tag{7.2}$$

The shape parameters $\alpha > 0$ and $n > 0$ may vary in space, reflecting the soil heterogeneity. Equations (7.1) and (7.2) are subject to initial and boundary conditions:

$$\begin{aligned} \theta(\xi,\zeta,0) &= \theta_\mathrm{init}, \\ \psi(\xi,\zeta=0,\tau) &= \psi_0, \\ \theta(\xi,\zeta\to\infty,\tau) &= \theta_\mathrm{init}, \\ \frac{\partial\psi}{\partial\xi}(\xi=\pm1,\zeta,\tau) &= 0. \end{aligned} \tag{7.3}$$

The numerical values for these are $\theta_\mathrm{init} = 0.2$ and $\psi_0 = 0.01\,\mathrm{m}$.

As noted earlier, the Richards equations is regarded as the reference model: we assume that solutions of the presented equations have no model error.

Model Parameterization

Among all the model parameters, saturated hydraulic conductivity K_s and soil parameter α vary most and exhibit the highest degree of uncertainty [127, 145, 154]. In line with this observation, we treat $K_\mathrm{s}(\xi,\zeta)$ and $\alpha(\xi,\zeta)$ as random fields, while assuming the remaining parameters to be constant (in space and time) with the following known values: $n = 1.81$, $\phi = 0.42$ and $\theta_\mathrm{i} = 0.13$. These values are the mean values provided by Russo and Bouton [127, Table 3a].

We assume that the natural logarithms of K_s and α are statistically independent, second-order stationary Gaussian random fields. The soil data

	μ	σ^2	λ_1[m]	λ_2[m]
$\ln K_s$	-3.58	0.89	0.7840	0.2123
$\ln \alpha$	-3.01	0.63	0.2554	0.1117

Table 7.1.: Statistical properties of K_s and α: mean μ, variance σ^2 and correlation lengths λ_1, λ_2 [127, Table 3a].

analyzed by Russo and Bouton [127] and various subsequent studies suggest the use of an anisotropic exponential covariance function

$$\begin{aligned} C(d_1, d_2) &= \sigma^2 \exp[-s], \\ s &= \sqrt{(d_1/\lambda_1)^2 + (d_2/\lambda_2)^2}, \end{aligned}$$

where σ^2 is the variance, d_1 and d_2 are the horizontal and vertical distances between two points, and λ_1 and λ_2 denote the horizontal and vertical correlation lengths. The specific field parameters used in this chapter are again taken from Russo and Bouton [127, Table 3a]. The values are summarized in Table 7.1.

Computation of the Quantity of Interest

Solving Eqs. (7.1)–(7.3) yields realizations of the state variable $\theta(\xi, \zeta, \tau)$. To compute the QoI (the total amount of infiltrated water), we first calculate the infiltration depth $\zeta_f(\xi)$ at each point in the domain. We are only interested in the state at time $\tau = 30$ min, so in the following we omit the time argument τ. The expression for the infiltration depth is

$$\zeta_f(\xi) = \int_0^\infty \frac{\theta(\xi, \zeta, 30) - \theta_{\text{init}}}{\phi - \theta_{\text{init}}} \, d\zeta.$$

Note that the infiltration depth is only a conceptual quantity. In reality, the water content along a vertical slice transitions smoothly from the fully saturated state ($\theta = \phi$) to the dry state ($\theta = \theta_{\text{init}}$). There is no sharp wetting front as shown in Figure 7.3. The infiltration depth ζ_f can be understood as the hypothetical position of the wetting front if there were a sharp front and if the area under both curves were the same.

From the infiltration depth, the total amount of infiltrated water Q follows as

$$Q = (\phi - \theta_{\text{init}}) \int_{-1}^{1} \zeta_f(\xi) \, d\xi. \tag{7.4}$$

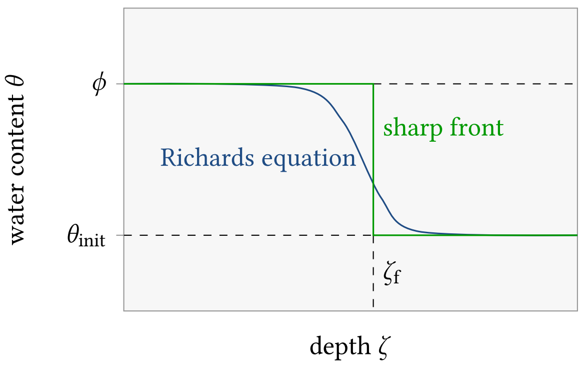

Figure 7.3.: The water content θ over a vertical slice in the domain: Richards equation (blue) versus a sharp front (green).

The considered problem is two dimensional, so the unit of Q is m^2.

7.1.2. Green-Ampt Model

The Green-Ampt model provides a simplified solution to the Richards equation for infiltration processes. It makes the following additional assumptions:

1. Soil parameters are homogeneous in the vertical direction.
2. Water flows only vertically, from the soil surface downwards, which means that there is no liquid exchange between different vertical slices in the domain. The Green-Ampt model itself is a one-dimensional model (over the coordinate ζ). To solve our two-dimensional infiltration problem, the model is evaluated separately for each vertical slice. As a consequence, the horizontal coordinate ξ does not occur in any of the following equations.
3. There is a sharp wetting front at depth ζ_{f}, which separates the dry soil ($\theta = \theta_{\text{init}}$) from the wet soil ($\theta = \phi$), such that

$$\theta(\zeta, \tau) = \begin{cases} \phi & \text{for } \zeta < \zeta_{\text{f}}(\tau) \\ \theta_{\text{init}} & \text{for } \zeta \geq \zeta_{\text{f}}(\tau) \,. \end{cases}$$

   This is exactly the sharp front shown in Figure 7.3.

For our infiltration problem, the Green-Ampt model has an implicit solution for the depth of the wetting front $\zeta_{\mathrm{f}}(\tau)$ [101, 154, 155]:

$$\zeta_{\mathrm{f}}(\tau) - (\psi_0 - \psi_{\mathrm{f}}) \ln\left(1 + \frac{\zeta_{\mathrm{f}}(\tau)}{\psi_0 - \psi_{\mathrm{f}}}\right) = \frac{K_{\mathrm{s}}}{\phi - \theta_{\mathrm{i}}}\tau, \tag{7.5}$$

where the pressure head at the infiltration front, ψ_{f}, is set to a capillary drive [101, 154],

$$\psi_{\mathrm{f}} = -\int_{\psi_{\mathrm{i}}}^{0} K_{\mathrm{r}}(\psi)\, \mathrm{d}\psi.$$

The pressure head in the dry soil, ψ_{i}, is related to the corresponding irreducible water content θ_{i} by the van Genuchten model [Eq. (7.2)].

As before, we are only interested in the state at $\tau = 30\,\mathrm{min}$. By setting $\tau = 30\,\mathrm{min}$ and solving Eq. (7.5) once for each vertical column, we obtain the wetting front as a function of the horizontal coordinate: $\zeta_{\mathrm{f}}(\xi)$. This wetting front has the same form as the prediction of the Richards equation, so the QoI can similarly be calculated using Eq. (7.4).

Model Parametrization

While the deterministic parameters from the Richards equation (n, ϕ and θ_{i}) can immediately be used in the Green-Ampt model, the two random fields for K_{s} and α require some pre-processing: due to the assumption of homogeneity in vertical direction, we have to replace the two-dimensional random fields for K_{s} and α by a collection of homogeneous one-dimensional isolated flow tubes, labeled by their horizontal position ξ. This parametrization is known as the Dagan-Bresler parametrization [37]. To obtains the K_{s} and α values for each column, we need to take vertical averages over the two-dimensional fields. A conceptual difficulty here is that the depth over which the average is to be taken is not known a priori. It would be desirable to average from the soil surface down to the wetting front (i. e., over the interval $[0, \zeta_{\mathrm{f}}]$), so that exactly those soil properties are included, that are actually involved in the infiltration process. The infiltration depth, of course, is not known before the calculations. Therefore, we use the following procedure to get an estimate of ζ_{f}. First, we draw a sample of K_{s} and α from their respective distributions. The underlying random fields are stationary, so the distribution is the same at any point in space. We then pretend that the drawn values were constant over the whole domain (even though, in reality the drawn values are point values in space), and insert these into the Green-Ampt solution in Eq. (7.5) to obtain an infiltration depth. We repeat this process until we obtain a large sample of infiltration depth estimates ζ_f. From this sample, we construct an empirical cumulative distribution function $F(\zeta) = P[\zeta_\mathrm{f} < \zeta]$. This function transitions from $F(0) = 0$ to $F(\zeta \to \infty) = 1$. Finally, the parameters K_s and α are averaged vertically using a weighted mean with weights proportional to $1 - F(\zeta)$. The conductivity K_s is averaged with a harmonic mean (analogous to multiple resistors in a series connection) [146], and the shape parameter α is averaged arithmetically.

7.1.3. Numerical Implementation

The Richards equation is solved using the USGS software package vs2dt, which can be downloaded at `http://water.usgs.gov/software/` (as of April 2016). Horizontally, the domain is discretized with 50 equally spaced cells. Vertically, it is discretized with 30 cells and a grid refinement towards the top boundary. While the conceptual domain $\mathbb{D}$ is infinitely deep, the numerical model is cut off at a depth of 2 m. Some preliminary calculations showed that, within the first 30 min, the infiltrated water would never reach a depth larger than this.

The Green-Ampt model is solved using the Matlab function `fzero`, which uses a combination of bisection, secant, and inverse quadratic interpolation methods. One simulation run solves Eq. (7.5) for all 50 soil columns defined by the discretization of the Richards equation.

7.1.4. Bayesian Updating

To compare the two models in a meaningful way and to isolate the effect of the model error, we can only assimilate data from quantities that can be predicted by both models. Soil-moisture sensors provide point-wise measurements of the water content $\theta(\xi, \zeta, \tau)$. While these data can be assimilated into predictions obtained with the Richards equation, the Green-Ampt model predicts only the infiltration depth $\zeta_\mathrm{f}(\xi)$. We therefore need to convert measurements of $\theta(\xi, \zeta, \tau)$ into "measurements" of $\zeta_\mathrm{f}(\xi)$, which means that the measurement quantity Z must be a function of $\zeta_\mathrm{f}(\xi)$. To this end, I developed the following simplified measurement model.

Let (ξ, ζ) denote a sensor's position. We model the sensor's output Z as

$$Z(\xi,\zeta) = \begin{cases} \text{dry} & \text{if } \zeta_\mathrm{f}(\xi) < \varepsilon \cdot \zeta \\ \text{wet} & \text{otherwise,} \end{cases} \tag{7.6}$$

where ε represents a measurement error, introducing random noise. This sensor model does only have the two possible measurement values "dry" and "wet". If it were $\varepsilon = 1$, this sensor model would represent a perfect sensor. We assume ε to be log-normally distributed with $\ln \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon)$, so that in some cases, the measurements can be wrong, especially if the sensor is very close to the wetting front ($\zeta \approx \zeta_\mathrm{f}(\xi)$). In our numerical examples, we set $\sigma_\varepsilon = 0.1$.

With the measurement model in place, we can now derive an expression for the likelihood term. Assume that a wetting front ζ_f is given (as a function over ξ). By solving the measurement model [Eq. (7.6)] for ε and inserting into the cumulative distribution function (cdf) of the normal distribution $F_\mathcal{N}$, we obtain the conditional probability of getting the measurement "wet" in terms of the sensor's position (ξ, ζ), conditional to the wetting front, as

$$P\Big[Z(\xi,\zeta) = \text{wet}\,\Big|\,\zeta_\mathrm{f}\Big] = F_\mathcal{N}\left(\sigma_\varepsilon^{-1} \ln \frac{\zeta_\mathrm{f}(\xi)}{\zeta}\right).$$

The conditional probability of measuring "dry" is, of course, the complement

$$P\Big[Z(\xi,\zeta) = \text{dry}\,\Big|\,\zeta_\mathrm{f}\Big] = 1 - P\Big[Z(\xi,\zeta) = \text{wet}\,\Big|\,\zeta_\mathrm{f}\Big].$$

If multiple sensors are present, we assume their measurement errors to be independent. The probability of measuring values at multiple sensors is then the product of the probabilities of the individual measurements. For a number of sensor positions $(\xi_1, \zeta_1), \ldots, (\xi_n, \zeta_n)$ and measurement values $z_1, \ldots, z_n$ with $z_i \in \{\text{dry}, \text{wet}\}$, we obtain

$$P\Big[Z(\xi_1,\zeta_1) = z_1, \ldots, Z(\xi_n,\zeta_n) = z_n\,\Big|\,\zeta_\mathrm{f}\Big] = \prod_{i=1}^{n} P\Big[Z(\xi_i,\zeta_i) = z_i\,\Big|\,\zeta_\mathrm{f}\Big].$$

Recall, that this term is the likelihood of the measurement $z_1, \ldots, z_n$, see Eq. (2.4) in Section 2.2.

All experiments are performed with virtual measurements. That means that one realization of the input parameters is selected as the virtual truth.

From this realization, the virtual data are generated via the measurement model in Eq. (7.6). In some cases, the whole procedure is repeated multiple times, selecting multiple truths and averaging the errors in order to obtain more stable and robust results.

Stochastic Discretization

As described earlier, the input uncertainty for the experiment is given in terms of the two random fields for K_s and α. To evaluate errors in a meaningful way, we need to have a reference solution with which all approximate solutions can be compared. Given the high stochastic dimensionality of K_s and α, such reference solution cannot be obtained exactly. We therefore replace the original random fields by a discrete distribution on a finite sample, which we call *base sample*. As a result, we obtain a slightly different problem, which we define as the reference. All subsequent calculations are done for the newly defined reference problem. The consequences of this approach are these:

First, we are able to calculate the reference solution exactly. To do so, we simply insert the full base sample into the model equations and solve for the QoI. The resulting sample of the QoI is the reference prior distribution of the QoI. A reference posterior distribution can be obtained by giving each realization a weight proportional to its likelihood, see Section 2.2.2.

Second, for applying the Monte-Carlo method to the problem, we have to draw realizations from the base sample *with replacement*. That way, we can obtain sample sizes that are larger than the base sample, and still observe the typical Monte-Carlo convergence behavior with a rate of $O\left(N^{-1/2}\right)$. With this property, our approach for error calculation is highly similar to the bootstrapping method [46]. The main difference is that in this experiment, we define the base sample as the reference, while the bootstrapping method uses the base sample to make statements about the original distribution, from which the base sample was drawn (in our case the two random fields).

In the experiment we draw a base sample of size 10^4. For the Monte-Carlo simulations we draw samples of sizes between $N = 1$ and $N = 10^6$.

7.2. Simulation Results

Recall that the QoI is the total amount of infiltrated water Q. The aim of all calculations is to find the (prior or posterior) distribution of Q. The reference solution can be obtained by inserting the full base sample into Richards equations. To measure the accuracy of any approximation, we use the earth movers distance (EMD) from the reference. As noted in Section 2.2.3, the EMD can be calculated directly because all distributions are given in form of samples. So in the following discussion, the term *error* always refers to the EMD between an approximation and the reference solution.

To emphasize the difference between the model error and stochastic error, I will use the following notation: all errors are written in the form $D^N_{M,n}$. The subscript $M \in \{\mathrm{R}, \mathrm{G}\}$ denotes the used model. It is either the Richards equation ($M = \mathrm{R}$) or the Green-Ampt model ($M = \mathrm{G}$). The second subscript n denotes the number of soil moisture sensors used ($n = 0$ for the prior). Finally, N is the number of realizations used in the Monte Carlo simulations. In the convergence analysis, N is varied from 1 to 10^6 and, according to the bootstrapping approach, samples are drawn with replacement. A special case are simulations with the full base sample *without* replacement: these calculations do not have a stochastic error, so the error represents the pure model error. Such cases are marked with the superscript star *. For example $D^\star_{\mathrm{G},0}$ denotes the prior model error of the Green-Ampt model.

7.2.1. Infiltration Depth

Figure 7.4 shows the spatial variability of the prior and posterior distributions of the infiltration depth $\zeta_\mathrm{f}(\xi)$ for one specific virtual truth. The posterior was calculated with data from seven equidistant moisture sensors. The figure also shows the virtual truth from which the data are generated via the measurement model, see Eq. (7.6). The plots were generated using the full base sample. Therefore, there is no stochastic error in these calculations, and all discrepancies between the two results are due to the model error.

The prior computed with the Green-Ampt model overestimates the infiltration depths on average by 0.005 m, which corresponds to the relative error of about 5%. The distribution's width is slightly underestimated. In the posterior distributions, the Green-Ampt model again underestimates

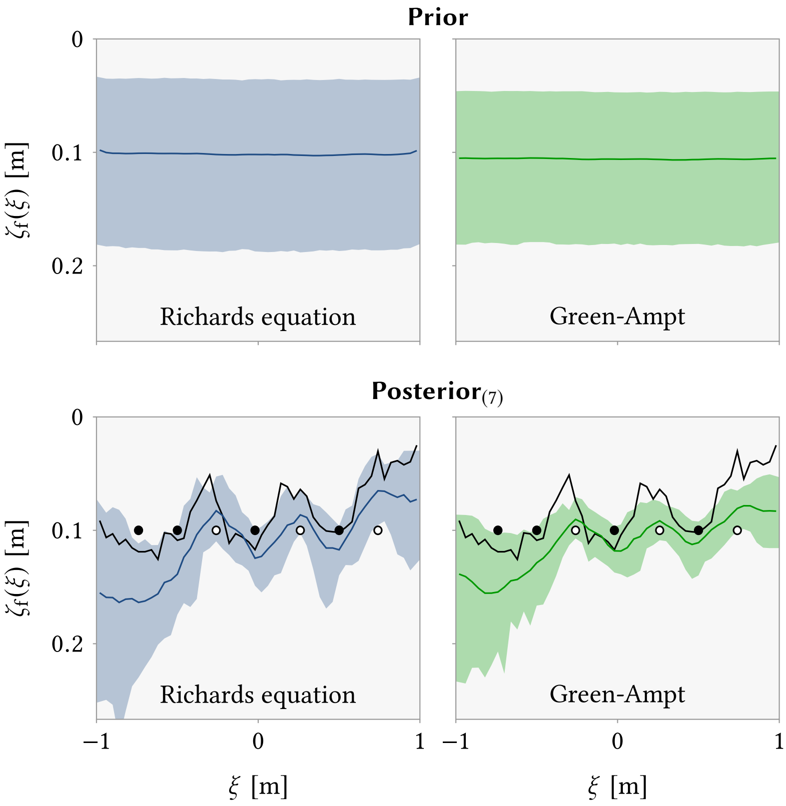

Figure 7.4.: Prior and posterior statistics of the infiltration depth $\zeta_f(\xi)$ at $\tau = 30$ min. Colored lines: ensemble mean. Shaded areas: pointwise 10th and 90th percentiles. Black line: virtual truth. Circles: moisture sensors, wet - black, dry - white. Both ζ_f and ξ are displayed in [m]. All diagrams show the same part of the domain.

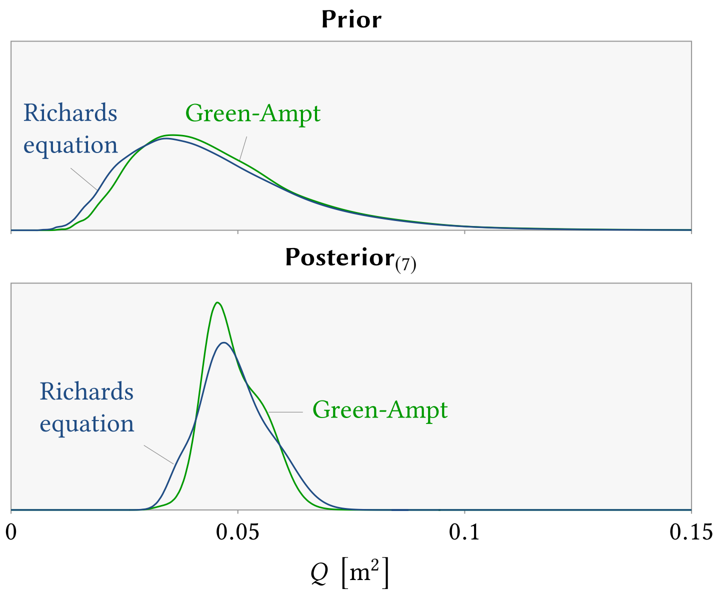

Figure 7.5.: Prior and posterior density estimates of the total amount of infiltrated water Q for the two models.

the distribution's width: the shaded area is smaller than its reference. The virtual truth leaves the shaded area in about one third of the domain.

Figure 7.5 shows density estimates (using the Matlab function `ksdensity`[1]) of the QoI Q, see Eq. (7.4). Again, the full base sample was used. This plot confirms the previous observations. In the prior, the reduced model slightly overestimates the water content. In the posterior, the means of both models almost align. For a quantitative comparison, the model errors of the Green-Ampt model are $D^{\star}_{\mathrm{G},0} = 1.87 \cdot 10^{-3}\,\mathrm{m}^2$ and $D^{\star}_{\mathrm{G},7} = 1.60 \cdot 10^{-3}\,\mathrm{m}^2$, which means that the posterior error is slightly smaller than the prior error.

[1]The kernel width was chosen automatically with the `support` option set to `positive`.

7.2.2. Computing Times and Convergence Behavior

We now return to the model-selection problem: given limited computational resources, which model has the smallest total error? We compare the total errors (model error plus stochastic error) of both models as a function of computing time.

We employ the setup with seven soil moisture sensors again and use the same virtual measurement values as in the previous section. Figure 7.6 shows the prior ($n = 0$) and posterior ($n = 7$) convergence plot of $D^N_{M,n}$ for both models ($M \in \{\mathrm{R}, \mathrm{G}\}$) in terms of the sample size (between $N = 1$ and $N = 10^6$). For each data point the error was averaged over 250 different random samples to ensure that the results are robust against sampling artifacts. One realization of the Richards equation takes about 56 s to compute, while one realization of the Green-Ampt model takes 0.087 s. This is a ratio of more than 600 : 1.

Since the Richards equation represents the reference model, its error converges to zero, $D^\star_{\mathrm{R},n} = 0$, by definition. The convergence rate is $\mathcal{O}(N^{-1/2})$, as expected. The error of the Green-Ampt model solution does not converge to zero, but to $D^\star_{\mathrm{G},0}$ and $D^\star_{\mathrm{G},7}$, respectively.

The model-selection problem can be solved directly from the convergence plot. For both the prior and the posterior there exists a computing time threshold T_n that marks the time after which the models should be switched. If the modeler has less than this time available, the Green-Ampt model should be used, otherwise the Richards equation yields better results. For the prior this threshold is $T_0 = 1.1 \cdot 10^4\,\mathrm{s} \approx 3\,\mathrm{h}$, for the posterior it is $T_7 = 6.3 \cdot 10^5\,\mathrm{s} \approx 174\,\mathrm{h}$. This means that if the available computation time is between T_0 and T_7, then the optimal model selection depends on whether the prior or the posterior is to be calculated. The availability of data favors the use of the reduced model.

A comparison of the two convergence plots in Figure 7.6 suggests that this result could be caused by two effects:

1. Available data reduce the model error, $D^\star_{\mathrm{G},7} < D^\star_{\mathrm{G},0}$.
2. Available data increase the stochastic error. While the asymptotic convergence behavior of the Richards equation is $C/\sqrt{N}$ in both cases, the posterior convergence has a larger multiplicative constant C and therefore reaches the same accuracy later than the prior does.

In the following two sections, we investigate these two effects in more detail.

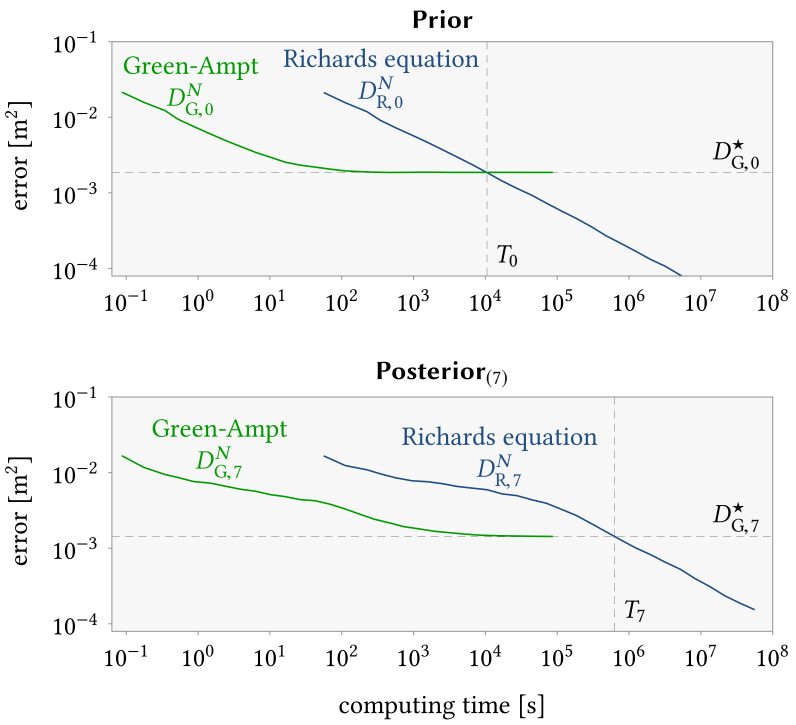

Figure 7.6.: Convergence of the prior and posterior distributions of the total amount of infiltrated water. Horizontal lines: model errors $D^\star_{G,0}$ and $D^\star_{G,7}$. Vertical lines: time thresholds T_0 and T_7.

7.2.3. Impact of Data on the Model Error

To check the extent to which measurements can reduce the model error, we vary the sampling density (i.e., the number of sensors n) and calculate the model error $D^{\star}_{\mathrm{G},n}$. One would expect the influence of the prior to diminish and the model error to decrease as the number of sensors increases. The horizontal positions of the sensors are selected via Hammersley sampling [62]. This sampling method allows us to spread out the sensors as equidistantly as possible while, at the same time, generating a nested set of sampling locations.

Figure 7.7 on shows the model error $D^{\star}_{\mathrm{G},n}$ as a function of the number of sensors $n = 1, \dots, 31$. The error is averaged over 100 repetitions, in which a different realization was selected to represent the virtual truth for generating the measurement data. One can see that the model error decreases at first until a minimum is reached with 3 sensors. Then the error gradually increases. The error with 7 sensors is on average larger than the error without measurements. That means that the decrease observed previously in Figure 7.6 was specific to the precise data used in that section and cannot be expected on average.

The increase in the model error for a large number of sensors shows that, among the base sample of 10^4 Green-Ampt solutions, there are no realizations that fully resemble the true wetting front. In the situation with data from more than 5 sensors, the model complexity of the Green-Ampt model is too low to keep up with the increasing number of sensors. The exact number of sensors that achieve a minimal model error is, of course, problem dependent. As a technical side node, a sampling density of 31 sensors on a domain of 2 m length is already unrealistically high for a real measurement campaign.

I conclude that the initial conjecture was incorrect: additional measurements do not, in general, lower the model error. The relationship between the number of sensors and the model error is a nonmonotonic function.

7.2.4. Impact of Data on the Stochastic Error

Finally, we investigate the extent to which the number of sensors affects the stochastic error. The upper plot in Figure 7.8 shows the convergence of the Monte Carlo solution with the Richards equation with 0 to 16 moisture sensors. At the right end of the plot, where the asymptotic convergence be-

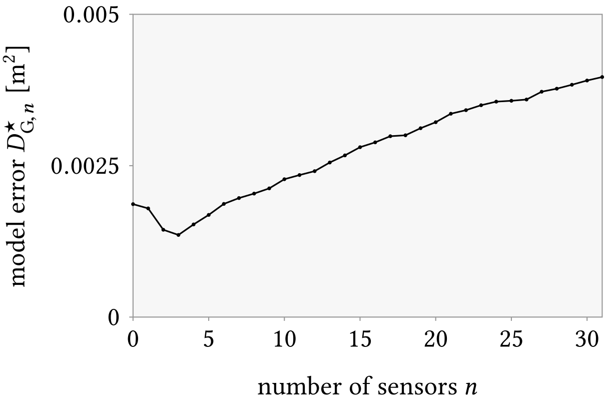

Figure 7.7.: Model error of the Green-Ampt model as a function of the number of sensors.

havior is attained, the individual data lines are perfectly ordered according to the number of sensors.

The lower plot in Figure 7.8 shows the rightmost data points from the upper plot (the data points for sample sizes of 10^6) as a function of the number of sensors. This figure confirms the previous observation that the stochastic error increases with the number of measurements.

7.3. An Approach to Model Selection

In this section, we recap the findings from the previous section and formulate a possible approach to the model-selection problem. Solving the model-selection problem is a matter of determining the time threshold T_n. Once it is known, the modeler can decide which model to use.

The convergence behavior shown in Figure 7.6 gives rise to two observations:

1. If a reduced-complexity model is much faster to solve than its high-fidelity counterpart, then, given sufficient computing time, the sampling error in the solution of the reduced-complexity model is negligible relative to its model error. In other words, the total error of the

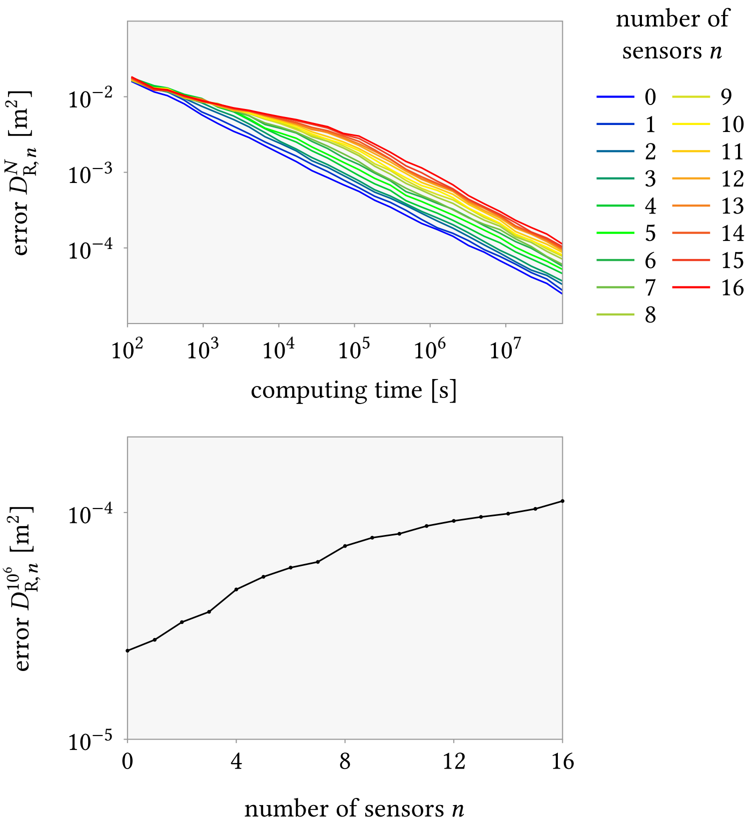

Figure 7.8.: Top: convergence of the Monte Carlo simulations with the Richards equation for different numbers of sensors. Since the Richards equation is the reference, the error shown is a pure sampling error. Bottom: the last data point from the upper plot ($D^{10^6}_{\mathrm{R},n}$), shown as a function of the number of sensors n.

reduced-complexity model at the time threshold T_n is constant and equal to $D^{\star}_{\mathrm{G},n}$.

2. The computing time T_n is sufficient to enable the high-fidelity model to reach the asymptotic convergence behavior of the form $C_n/\sqrt{N}$.

Let τ_{s} denote the computing time necessary to evaluate the high-fidelity model once. Then, the time threshold T_n is found by equating the two errors, $D^{\star}_{\mathrm{G},n} = C_n/\sqrt{N}$, which yields

$$T_n = \tau_{\mathrm{s}} N = \tau_{\mathrm{s}} \left(\frac{C_n}{D^{\star}_{\mathrm{G},n}} \right)^2 . \tag{7.7}$$

This general result holds for any QoI and any error measure, as long as the QoI estimate converges with the rate of $\mathcal{O}(N^{-1/2})$.

Figure 7.9 shows the dependence of the computing time threshold T_n estimated with Eq. (7.7) on the number of sensors n. The increase in the sampling error, quantified by the factor C_n (Fig. 7.8), outweighs the increase in the model error $D^{\star}_{\mathrm{G},n}$ (Fig. 7.7), such that the time threshold T_n increases with the number of sensors n. These results are averaged over the data from 100 different virtual truths. Therefore, the effect is not as strong as in the example given in Section 7.2.2, which represented a single realization of the virtual truth and by coincidence had a stronger effect.

Equation (7.7) reveals the difficulty in solving the model-selection problem a priori. To do so, one needs to determine both the model error of the reduced model, $D^{\star}_{\mathrm{G},n}$, and the multiplicative constant C_n in the convergence behavior of the complex model. These two quantities depend on the amount of available measurements n, as shown in the previous two sections.

The constant C_n could possibly be estimated using the reduced model if one assumes that the reduced model converges to its limit with the same constant as the complex model does (in terms of number of realizations, not in terms of computing time). For the estimation of the model error $D^{\star}_{\mathrm{G},n}$, however, I am not able to provide a general approach: while in the absence of measurements one could compare a small number of realizations of both models to get an estimate of $D^{\star}_{\mathrm{G},n}$, the presence of measurement data changes the model error; this change is nonmonotonic in the amount of available data (Fig. 7.7). Overall, these factors make it difficult if not impossible to get an a priori estimate of the time threshold T_n.

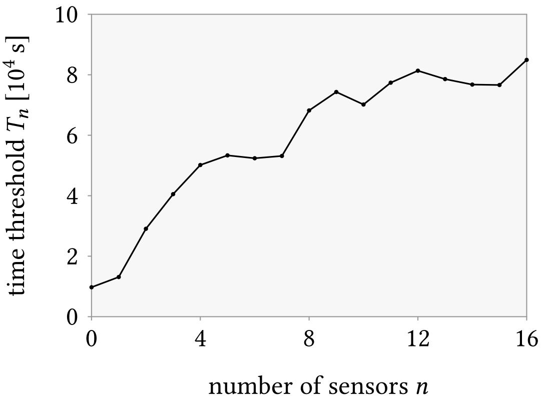

Figure 7.9.: Dependence of the time threshold T_n, see Eq. (7.7), on the number of sensors n.

7.4. Conclusions

In this chapter I studied the interplay between Bayesian updating and the model-selection problem in multi-fidelity modeling. The study is centered around an exemplary infiltration problem, which can be solved by either the Richards equation or the Green-Ampt model.

A modeler can often choose between an accurate, but slow model (the reference model) and a faster, but less accurate model (the reduced model). This choice is especially interesting in stochastic computations which require repeated evaluations of the model function. If the computing time available is limited, then the faster model allows to evaluate the model function more often, leading to a smaller stochastic error (e. g., Monte-Carlo error). Overall it is not a priori clear which of the two models results in the smaller total error (= model error plus stochastic error). These considerations, of course, are all under the premise that the modeler wants to use only one of the two models. There are also approaches, such as MLMC, that use both models together to obtain even more accurate results. These are not considered here.

Our study showed that in a very tight time constraint, the stochastic error dominates the total error and therefore the reduced model tends to perform better. At the other extreme, without a time constraint, the reference model, of course, leads to a smaller total error. Between these two extremes there exists a time threshold T that marks the time at which the model should be switched. This time threshold answers the model-selection problem: if the modeler has less time than T, then the use of the reference model is not justified and the reduced model should be used. If more time than T is available, the reference model leads to the overall better results.

The availability of measurement data for Bayesian updating has an influence on the model error of the reduced model and on the stochastic error of both models. In this study we found that, indeed both types of errors change when data become available. Overall, the errors change in favor of the reduced model. Within the tested range, the time threshold T increases almost monotonically with the number of measurements. This increase is rather drastic, changing by a factor of 8.

Furthermore, the study showed that it is difficult if not impossible to calculate the time threshold T a priori.

Overall, our analysis suggests that the availability of data favors the use of reduced models. However, this study does not lead to a practical guideline for the model-selection problem because of the difficulty in calculating the time threshold T. In addition, all results are obtained from one specific infiltration problem, so the generality of these findings for other phenomena remains an open question.

Part III.

Epilogue

8. Summary, Conclusions and Outlook

In this final chapter, I will summarize the findings from the previous four chapters, discuss these findings to formulate overarching conclusions, and present follow-up questions for future research.

8.1. Summary

In this thesis, I considered uncertainty quantification (UQ) problems under time constraints. If we know in advance that the computing time is limited, then we are interested in finding the best result possible in the time available. This is a question of optimality, so I approached the construction of numerical methods via mathematical optimization. This thesis provides four contributions:

1. In Chapter 4, I developed a sampling rule for the optimal construction of polynomial response surfaces. Such response surfaces are commonly used in UQ within the methodology of stochastic collocation for polynomial chaos expansions. The sampling rule I developed is called optimized stochastic collocation (OSC). With OSC, optimal nodes and weights are found by mathematical optimization of an operator norm.

   In a numerical experiment, I compared the performance of OSC with other commonly used sampling rules, such as tensor grids, random sampling and PCM. Among the sampling rules compared, OSC turned out to be the most efficient one because it is stable, flexible and versatile. All other sampling methods lack at least one of these properties. Stability guarantees that the response surface will not oscillate between the nodes, flexibility allows the modeler to choose the number of function evaluations freely, and versatility means that the method can handle multivariate input distributions with statistical dependence.

2. In Chapter 5, I investigated under what conditions optimal response surface methods for uncertainty propagation exist. As opposed to the approach in Chapter 4, I formulated this as an optimization problem to determine both the optimal nodes and the optimal functional form of the response surface. I compared three different objective functions for the optimization: the Lebesgue constant, an operator norm and the expected squared error.

   It turned out that only the third objective function leads to a practical and useful response surface method. This method requires the modeler to describe the unknown model function as a random field and, as I was able prove, the optimal response surface method coincides with the Kriging method that is well-known in geostatistics.

   This analysis showed that an optimal response surface method under time constraints can only exist if the modeler makes additional assumptions about the shape of the model function.

3. In Chapter 6, I developed a sequential design for Bayesian inverse problems. Bayesian inverse problems differ from forward uncertainty quantification problems in that the region of interest in the parameter domain is not known beforehand. For this reason, I chose a sequential design (of computer experiments) for the selection of nodes. Nodes are chosen one at a time. The model response at all previous nodes is used to determine the new point that promises the most benefit for the overall goal of solving an inverse problem. The sampling method relies on an explicit model for code uncertainty which is realized via a random field.

   Numerical experiments showed that the sequential design is able to identify the important parts of the parameter domain quickly, and so achieves a better efficiency than non-sequential methods. The method is able to handle both Gaussian and non-Gaussian random fields. A simple one-dimensional example showed that, with this property, the sequential design is generally able to handle discontinuous model functions with uncertain jump locations.

4. In Chapter 7, I investigated the use of low-fidelity models for Bayesian updating. Whether a slow (but accurate) or a fast (but inaccurate) model is preferable, is a problem of model selection. Classical model selection methods only compare models in terms of their accuracy,

but do not consider computational costs. In UQ problems, time constraints have an important influence on the model selection problem because, in a given time, a slow model can only be evaluated a few times, and this leads to a large stochastic error (e.g. Monte-Carlo error). Therefore, the choice between a high-fidelity model and a low-fidelity model is a trade-off between model error and stochastic error.

For such trade-off situations, I investigated what influence the amount of available data in Bayesian updating has on the optimal model selection. It turned out that the amount of data has an influence on both the model error and the stochastic error and therefore also influences the optimal model selection. In the numerical example, the general tendency was that a larger amount of data favors the use of the reduced model.

Overall, these four chapters make valuable contributions to uncertainty quantification in practical applications. All the methods I developed consider time constraints right from the start, and so allow the practitioner to make efficient use of the time available.

8.2. Discussion and Conclusions

I will now discuss the results presented in the thesis in a broader context and draw overarching conclusions. Altogether, there are four important conclusions I would like to formulate. The first three of these are directly related to the three hypotheses I proposed in the introduction in Section 1.4. Recall that the hypotheses were the following.

1. Under time constraints, code uncertainty plays an important role. It should be taken into account explicitly, for example by using emulators.
2. Under time constraints, optimization is a viable approach to surrogate modeling. Optimal methods are flexible automatically.
3. Under time constraints, all available information about the model function should be used.

I use O'Hagan's definition of the term "emulator": it denotes a random field used as a surrogate [109].

Do not create models, falsify models!

In the commentary paper “Popper, Bayes and the inverse problem” [143], Albert Tarantola writes:

Data are not to be used to create a model, but, instead, to falsify models.

Tarantola attributes the core idea of this suggestion to Karl Popper and therefore calls it “a modified version of the Popperian paradigm”. More precisely, he suggests the following steps for solving inverse problems.

1. Using everything known about the system beforehand (the a priori information), create a collection of all possible models for the system. These might be infinitely many.
2. For each model, solve the forward problem and make a prediction about the data.
3. Compare the observed data with the predictions and, according to some rule, reject all those models that did not predict the observations well enough.
4. The solution to the inverse problem consists of all models that were not rejected in the previous step.

Note that this solution consists of possibly infinitely many models. In contrast, many methods for inverse problems and calibration try to create a model based on the observed data, for example by a least-squares fit. The solution in such methods consists of only the one model that is believed to be the best explanation for the observation.

Now let us regard the handling of an expensive model function under time constraints as an inverse problem. Due to code uncertainty, we do not know the input-output relationship of this function. The best we can do is observe the model response at a finite number of model inputs. If we understand these observations as “data” about the model function, then finding the model function itself is an inverse problem. For this kind of inverse problem, Tarantola essentially suggests that we should use emulators and not response surface methods.

- A response surface method creates a model.
- An emulator is a collection of possible models, and when an emulator is conditioned to observations, some of these models are falsified.

Emulators do have the concept of code uncertainty built into them, while response surfaces do not. If we want to follow Tarantola's suggestion, we have to take code uncertainty into account.

Looking back at Chapters 5 and 6, we can see that Tarantola's suggestion is not just a philosophical argument, but actually has direct practical consequences and can lead to better numerical methods. In Chapter 5, where I derived optimal response surface methods, the only approach that actually resulted in a useful method was the approach via emulators. I would even argue that it would be best not only to use the resulting response surface (the field mean), but the conditioned emulator as a whole. This is also in line with Tarantola's suggested procedure, where the solution of an inverse problem consists of a collection of models. Then, in Chapter 6, the whole concept of a sequential design hinges on an explicit model for code uncertainty, which is realized by describing the model function as a random field. Furthermore, the likelihood estimator presented in that chapter includes the uncertainty in the random fields, so that at any point in time, the iterative sampling procedure can be stopped and the result will be a meaningful posterior distribution.

Intuitively, one might think that an emulator is a more blurry representation of a function and therefore makes less informative predictions. That is indeed true in the following sense: if the true model function u is the mean of the random field U, $u = \mathrm{E}_U [U]$ and we compare the two random variables $Y = u(X)$ and $Y' = U(X)$, then, by the law of total variance, Y' has a strictly larger variance than Y, as long as U has a positive variance. It may seem that a larger predicted variance is undesirable because it makes a prediction less informative. I argue that this larger variance is, in fact, desirable because it is a more correct representation of the available knowledge. As long as we have not evaluated the model function in all points, there will be code uncertainty. In the worst case, replacing the uncertain model function by a response surface can produce misleading results in the sense that the image of the response surface does not even contain the true model output. Such a response surface would be inappropriate for uncertainty quantification because it would claim that the actual real-world behavior of the system under consideration would be impossible and would assign it a probability (and probability density) of zero. An example for this effect is shown in Figure 6.1 in Chapter 6. There, the response-surface-based likelihood (the plug-in likelihood) is overly confident and biased, while the emulator-based likelihood (the best estimator) provides an adequate representation of the knowledge about the input parameter. Overall, I suggest modelers to take care when using potentially misleading UQ methods.

Of course, even if the general idea of using emulators is conceptually very clean, it still has practical problems: how do we specify an emulator such that the unknown model function is a realization of it? If we fail in this step and the model function is not a realization of the field, then all results might again be misleading.

Finally, I would like to emphasize that time constraints make an essential difference in this discussion. If we do not have time constraints and we are interested in the convergence behavior of a method, then using response surface methods is a viable approach. If a sequence of response surfaces converges to the model function, then also the degree to which the response surfaces are misleading will decrease. In a convergence analysis, it is not a problem if intermediate response surfaces are misleading.

To sum up, the findings in my work and the “modified version of the Popperian paradigm” as formulated by Tarantola [143] both support the first of my three hypotheses: code uncertainty should be taken into account explicitly. When working in UQ under time constraints, I recommend the use of emulators over the use of response surfaces. In convergence-based analyses, response surfaces, of course, are valuable tools.

Optimization is a viable approach to surrogate modeling and flexibility follows naturally.

All four of my contributions demonstrate that surrogate modeling can be successfully approached via optimization. This was my second hypothesis. Moreover, the resulting methods are automatically flexible, in the sense that the modeler can easily adapt the run time of a method to the time available. This is because the number of function evaluations enters the optimization process as part of the problem setting instead of coming out of the optimization as a result. Both OSC in Chapter 4 and the Kriging-like method presented in Chapter 5 allow the modeler to select the number of function evaluations n freely. The sequential design strategy presented in Chapter 6 is flexible by its sequential nature. The iterative procedure can be stopped at any desired time and, as I argued in that chapter, the best-estimate likelihood in any iteration is a valid solution to the inverse problem if code uncertainty is considered. Finally, the model selection problem in Chapter 7 has the concept of flexibility built into the problem description: if a certain amount of time is available, the goal was to select the model that allows us to use this time most efficiently.

All available information about the model function should be used.

My third hypothesis states that, under time constraints, all available information about the model function should be used. The results of Chapters 5 and 6 actually lead to an even stronger conclusion: if we pursue optimality instead of convergence, then additional assumptions about the model function are not just desirable, they are *essential*.

It is often possible to prove the convergence of a method for a relatively large class of model functions. For example, under many probability distributions, polynomials are dense in L^2 [49]. Therefore, it is possible to approximate any function in L^2 arbitrarily accurately by polynomials and we do not need to make smoothness assumptions about the function.

Optimality, in contrast, can only be achieved if we make additional assumptions about the model function. The two theorems in Chapter 5 are basically statements about the relationship between possible objective functions and the assumptions required to formulate them. For example, if the expected squared error is used as an objective function, then the model function has to be described by a random field. And as shown by the second theorem in Chapter 5, the performance of a response surface method depends directly on the correlation function of the random field. If the model response is assumed to be completely uncorrelated, then a model evaluation will not provide any useful information about the behavior of the model response elsewhere, and response surface modeling will not make sense. Therefore, we have to make the assumption that the random field has a non-zero correlation. Such an assumption is the price we have to pay if we aim for optimality.

Furthermore, my results showed that the achievable performance of a response surface method is “monotonous” with respect to the “strength of the assumptions”: a random field with a large correlation length will generally achieve a better value in the objective function than a field with a small correlation length. The same general behavior was found in the results in Chapter 6. There, I compared various random fields for the description of the response of the heat equation. The fields with stronger smoothness assumptions generally achieved a better accuracy in solving the inverse problem. The additional assumptions, of course, have to be valid. As I already mentioned in the previous section, the model function must be a realization of the used random field.

In the field of optimization, the so-called No Free Lunch theorems are well known [164]. They state that, on average over all possible optimization problems, all optimization algorithms perform equally well. An optimization algorithm can only perform better than a random search if additional knowledge about the optimization problem is available and used correctly. The same holds true for surrogate modeling: we can only create good surrogate models if we have additional knowledge about the model function and make proper use of it. The achievable accuracy depends directly on the amount of available knowledge.

Model selection with and without time constraints is entirely different.

For a fourth conclusion, it is interesting to compare the model selection problem under time constraints, see Chapter 7, with classical model selection approaches.

Most modelers agree that complex models should only be used if enough data are available because otherwise one runs the danger of overfitting the model. In other words, the presence of data favors the use of more complex models. This viewpoint only considers the predictive power of a model, but not its computational costs.

In contrast, under time constraints, the reverse can be true: in the numerical example in Chapter 7, I found that the availability of data in fact favors the use of the simpler model.

We can learn two things here: first, it makes a big difference whether we work under time constraints or not. In our case, the presence of time constraints even turned some common knowledge into the opposite. Second, when using a simple rule of thumb such as “the presence of data favors the use of more complex models”, one should have a clear understanding of what it means and when it applies.

Summary

In summary, my main four conclusions are these:

- under time constraints, emulators should be preferred over response surface methods;
- optimization is a viable approach to surrogate modeling and leads to flexible methods;
- if we aim for optimality instead of convergence, we have to make additional assumptions about the model function;
- model selection methods should consider time constraints right from the start.

8.3. Outlook

In this final section, I present a couple of ideas for future research.

- I believe it is worthwhile to follow the idea of the "modified Popperian paradigm" more consistently. After all, the main idea of UQ is to admit that certain information is not known precisely. It would be foolish to claim that a simulation coincides with reality. However, by using a response surface as a UQ method, we might fall into the same trap. It would be equally foolish to claim that the response surface coincides with the simulation. For this reason, I think that emulators with their explicit notion of code uncertainty have a great advantage over plain response surfaces. In future work, I would like to see more UQ methods that take code uncertainty into account and that make direct use of the additional information an emulator provides. This idea is directly connected to the notion of *overconfidence* which I think should also be explored more in the future.
- In the end of Chapter 4, I briefly outlined variations of the OSC method that are more directly adapted to the task of creating a response surface than OSC, but that lead to more difficult optimization problems. As a follow-up project, it would be interesting to investigate whether these methods are worth the additional effort and whether they actually result in better response surfaces.
  In this context, it is also interesting to investigate which of the existing approaches (quadrature and least squares) leads to better response surfaces.

- The sequential design in Chapter 6 is centered around the minimization of the squared error in the likelihood estimate, even though the overall goal is to achieve a small error in the posterior. It still remains open, whether an optimization of the error in the posterior (for example via the KL divergence) would result in an even better sampling method.
- In Chapter 7, I briefly commented on the differences between the model selection problem under time constraints and the classical one. In future work, these two different approaches should be combined. While my approach made the possibly wrong assumption that the reference model is known beforehand, the classical approach neglects computational costs altogether. In future work, the aspect of computational costs should be incorporated into Bayesian model averaging (BMA) methods. As my analysis showed, computational costs can result in additional uncertainty, so it seems to me that BMA methods already "speak the right language" to take computational costs into account.
- In the introduction, I argued that, optimal surrogate methods are more relevant for engineering practice than convergent methods. My reasoning was that, under time constraints, the computational effort cannot be increased arbitrarily. Aside from this practical reasoning, I believe that optimality is a stronger property than convergence in the sense that a sequence of optimal methods is automatically convergent. I do not have a formal proof for this conjecture so, in future work, it would be interesting to further explore the relationship between optimality and convergence.
- And last, I suggest a possible combination of emulators and low-fidelity models. Having a low-fidelity model available, we could write the reference model as the low-fidelity model plus a model error: $u_{\text{ref}} = u_{\text{low}} + \varepsilon_{\text{model}}$. If we now describe all three terms as emulators, then the emulator of the reference and the emulator of the low-fidelity model will have a non-zero cross-covariance. Using this cross-covariance, the emulator of the reference model can now be conditioned to function evaluations of the low-fidelity model. We then might be able to design a mixed sampling strategy that combines samples from both the reference model and the low-fidelity model to reduce the uncertainty about the reference model.

  In a Bayesian inverse problem as shown in Chapter 6, for example, it might be possible to use a space-filling design with the low-fidelity model first to explore the parameter domain and identify the high-likelihood regions. Then, the reference model could be used to further refine the most important part of the domain. Ideally, such a strategy could be realized as a sequential design with an objective function that takes into account both the expected information gain of each evaluation and the different computational costs of the two models. Such a method would also be able to decide when to use which of the two models, so that the modeler does not have to divide the overall procedure manually into an exploration phase and a refinement phase. Such phases should emerge automatically from the optimization: at early times, the reference model is highly uncertain and each evaluation of the low-fidelity model helps reduce this uncertainty at low computational costs. But later, when the uncertainty in the reference model becomes smaller, the possible contribution of the low-fidelity model diminishes and a further reduction of the uncertainty can only be achieved by using the reference model itself.

The approach of decomposing the overall response u_{ref} into multiple smaller parts is similar to multi-level Monte-Carlo methods.

A. Beyond Quadrature and Least Squares: Preliminary Results

In this Appendix, I present preliminary results for the generalization of OSC presented in Section 4.4. These results were obtained using the Lebesgue constant as an objective function as shown in that section. Under some additional assumptions, the Lebesgue constant can be calculated relatively easily. Recall that the Lebesgue constant is the smallest number $\Lambda\left(\mathcal{P}_{(\boldsymbol{x},A)}\right)$, such that

$$\left\|u - \mathcal{P}_{(\boldsymbol{x},A)}u\right\| \leq \left(1 + \Lambda\left(\mathcal{P}_{(\boldsymbol{x},A)}\right)\right)\left\|u - \mathcal{P}^{\star}u\right\|$$

for all $u \in \mathbb{T}$. If $u \in \mathbb{P}$, then it is $\mathcal{P}^{\star}u = u$ and so the right-hand side becomes zero. The Lebesgue constant can only be finite if, for all $u \in \mathbb{P}$, also $\mathcal{P}_{(\boldsymbol{x},A)}u = u$. If the dimension of $\mathbb{P}$ is the same as the number of nodes ($p = n$), then it follows that $\mathcal{P}_{(\boldsymbol{x},A)}$ has to be the interpolation operator on the nodes $\boldsymbol{x}$. In other words, if $p = n$, then the optimal matrix A is already uniquely defined. This reduces the dimension of the optimization problem because we only have to find the optimal nodes $\boldsymbol{x}$. The following results are obtained using this shortcut.

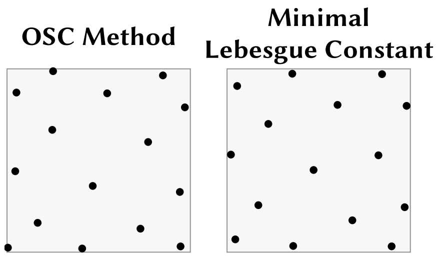

Figure A.1.: Nodes generated by a generalization of OSC.

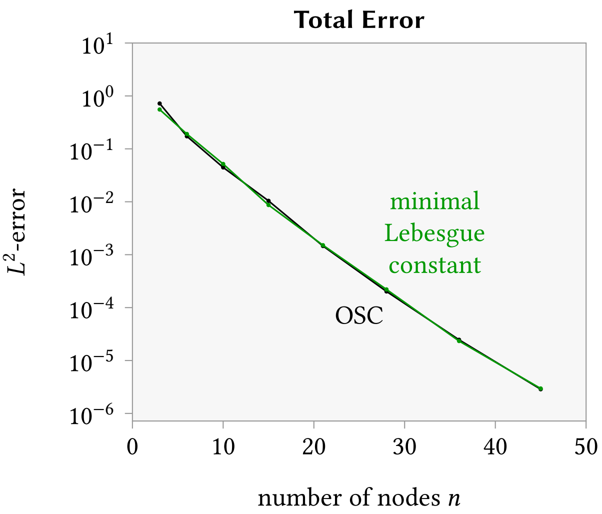

Figure A.2.: Total error for the approximation of the function $u(x_1, x_2) = \exp(x_1 + x_2)$ by polynomials of degrees between 1 and 8 over the uniform distribution on $[-1, 1]^2$. OSC in comparison with its generalization.

Figure A.1 shows the optimal sampling rules of sample size 15 for a uniform distribution on the domain $[-1, 1]^2$. The left diagram shows the result of the OSC method (this sample was already shown in Figure 4.1 in Chapter 4) and the right diagram shows the result of the generalization via the Lebesgue constant. While the precise node locations are different in both samples, they have in common that the points are most dense around the edges.

Figure A.2 shows the error behavior of the two approachs in comparison. The underlying experiment is the same as presented in Section 4.3.1 under the subheading “Increasing Polynomial Degree“: The parameter is uniformly distributed on $[-1, 1]^2$ and the model function is $u(x_1, x_2) = \exp(x_1 + x_2)$. The polynomial degree is increased together with the number of nodes. As shown in the plot, both methods have the same convergence behavior for this problem. Whether there are problems in which these two methods differ, is an open question for future research.

B. The Refinement Criterion in the Jump-Detection Problem

In this Appendix, I will show the derivation of the refinement criterion used in the Jump-Detection Problem, see Section 6.2.3.

We start with the second simplification described in Section 6.1.5, see Eq. (6.9). As noted there, this simplification is also possible for non-Gaussian emulators. Applying the simplification directly, leads to the expression

$$\hat{r}_n(\hat{x}) = C - \mathbb{E}_X \mathrm{Var}_{\hat{Y}} \mathbb{E}_{U_n^{\hat{x},\hat{Y}}} \left[L\left(X | U_n^{\hat{x},\hat{Y}} \right) \right], \tag{B.1}$$

where C is a constant that is independent from $\hat{x}$. The shape of the model function is fully defined by the jump location T. Therefore, we replace the expected value over $U_n^{\hat{x},\hat{Y}}$ by an expected value over T and keep in mind that the distribution of T depends on the value of $\hat{Y}$ and on the model response of the n function evaluations. The chain of operators becomes

$$\mathbb{E}_X \mathrm{Var}_{\hat{Y}} \mathbb{E}_T \left[L(X|T) \right].$$

For now, we ignore the expected value operator with respect to X and assume a fixed value $X = x$. Later, the resulting expression will be inserted into the corresponding integral over the values of X. The following expressions are all functions of x. Next, we derive the distribution of $\hat{Y}$. Like in Section 6.2.3, we define the two nodes that are closest to the jump:

$$\begin{aligned} a &= \max\{x_i | u(x_i) = 0\} \\ b &= \min\{x_i | u(x_i) = 1\}, \end{aligned}$$

and define the normalization function

$$h_{ab}(x') = \begin{cases} 0 & \text{if } x' < a \\ \frac{x'-a}{b-a} & \text{if } x' \in [a,b] \\ 1 & \text{if } x' > b. \end{cases}$$

Since the prior of T is $\mathcal{U}([0,1])$ and the only constraints are given by a and b, the conditional distribution of T after the first n model evaluations is $\mathcal{U}([a,b])$ and it follows, that the cdf of T is h_{ab}. The random variable $\hat{Y}$ is 1 if $T < \hat{x}$, so the distribution of $\hat{Y}$ follows from the cdf of T:

$$\hat{Y} = \begin{cases} 1 & \text{with probability } h_{ab}(\hat{x}) \\ 0 & \text{with probability } 1 - h_{ab}(\hat{x}) . \end{cases}$$

The random variable $\hat{Y}$ can be seen as a Bernoulli trial with probability $p = h_{ab}(\hat{x})$. As the variable $\hat{Y}$ does only take two possible values, also the inner term $\mathbb{E}_T[L(x|T)]$ can only take one of two values, which we call v_0 (for $\hat{Y} = 0$) and v_1 (for $\hat{Y} = 1$). Both v_0 and v_1 are still functions of x and $\hat{x}$. With these two values, we can calculate the variance as $\mathrm{Var}_{\hat{Y}}\mathbb{E}_T[L(x|T)] = (v_1 - v_0)^2 p(1-p)$. This is the variance of a standard Bernoulli trial $p(1-p)$ [158] multiplied with the squared spread of the distribution $(v_1 - v_0)^2$ to account for scaling. In the following we derive the values of v_0 and v_1 as functions of x and $\hat{x}$.

If the new node $\hat{x}$ is not in the interval $[a,b]$, then $\hat{Y}$ is not random and the variance term is zero anyways (because either $p = 0$ or $1-p = 0$). Therefore, the values of v_0 and v_1 are not relevant in this case. In the following, we assume $\hat{x} \in [a,b]$. The distribution of T conditional to the first n evaluations *and* the value of $\hat{Y}$ is the following:

- If $\hat{Y} = 1$, then $T \sim \mathcal{U}([a,\hat{x}])$
- If $\hat{Y} = 0$, then $T \sim \mathcal{U}([\hat{x},b])$

The expected value with respect to T has to be calculated according to these two cases. Similar to X we now assume a fixed value $T = t$ and insert the result into an integral over the values of T.

Finally we can have a look at the innermost term, the likelihood $L(x|t)$. Its value depends on x and t, as well as the measured value Z (which is either 0 or 1). Recall, that we assume a measurement model without measurement errors, so

$$L(x|t) = \begin{cases} 1 & \text{if } Z = 0 \text{ and } x < t \\ 0 & \text{if } Z = 0 \text{ and } x < t \\ 0 & \text{if } Z = 1 \text{ and } x \geq t \\ 1 & \text{if } Z = 1 \text{ and } x \geq t. \end{cases}$$

Now, turning back to the expected value over T, we find

- If $Z = 0$, then $v_0 = h_{\hat{x}b}(x)$, $v_1 = h_{a\hat{x}}(x)$
- If $Z = 0$, then $v_0 = 1 - h_{\hat{x}b}(x)$, $v_1 = 1 - h_{a\hat{x}}(x)$

In both cases, we obtain $(v_1 - v_0)^2 = (h_{\hat{x}b}(x) - h_{a\hat{x}}(x))^2$, which means that the resulting refinement criterion is independent from the observation Z.

Inserting into the variance term $(v_1 - v_0)^2 p(1-p)$ and integrating over x, we find

$$\hat{r}_n(\hat{x}) = C - h_{ab}(\hat{x})\left[1 - h_{ab}(\hat{x})\right] \cdot \int_0^1 \left(h_{\hat{x}b}(x) - h_{a\hat{x}}(x)\right)^2 \,\mathrm{d}x.$$

Even though the two functions $h_{\hat{x}b}$ and $h_{a\hat{x}}$ are dependent on $\hat{x}$, it turns out that the integral is constant:

$$\int_0^1 \left(h_{\hat{x}b}(x) - h_{a\hat{x}}(x)\right)^2 \,\mathrm{d}x = \frac{1}{3}(b-a),$$

so we finally obtain

$$\hat{r}_n(\hat{x}) = C - \frac{1}{3}(b-a) \cdot h_{ab}(\hat{x})\left[1 - h_{ab}(\hat{x})\right].$$

Bibliography

[1] R. Ababou, A. C. Bagtzoglou, and E. F. Wood, *On the condition number of covariance matrices in kriging, estimation, and simulation of random fields*, Mathematical Geology, 26 (1994), pp. 99–133.

[2] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions With Formulas, Graphs and Mathematical Tables*, Dover, New York, 10th ed., 1972.

[3] K. Aho, D. Derryberry, and T. Peterson, *Model selection for ecologists: the worldviews of AIC and BIC*, Ecology, 95 (2014), pp. 631–636.

[4] A. B. Antognini and M. Zagoraiou, *Exact optimal designs for computer experiments via Kriging metamodelling*, Journal of Statistical Planning and Inference, 140 (2010), pp. 2607–2617.

[5] I. Babuška, F. Nobile, and R. Tempone, *A stochastic collocation method for elliptic partial differential equations with random input data*, SIAM Journal on Numerical Analysis, 45 (2007), pp. 1005–1034.

[6] M. Balesdent, J. Morio, and J. Marzat, *Kriging-based adaptive importance sampling algorithms for rare event estimation*, Structural Safety, 44 (2013), pp. 1–10.

[7] V. Barthelmann, E. Novak, and K. Ritter, *High dimensional polynomial interpolation on sparse grids*, Advances in Computational Mathematics, 12 (2000), pp. 273–288.

[8] L. S. Bastos and A. O'Hagan, *Diagnostics for Gaussian process emulators*, Technometrics, 51 (2009), pp. 425–438.

[9] T. Bayes, *An essay towards solving a problem in the doctrine of chances (1763)*, republished in Biometrika, 45 (1958), pp. 296–315.

[10] J. Bect, D. Ginsbourger, L. Li, V. Picheny, and E. Vazquez, *Sequential design of computer experiments for the estimation of a probability of failure*, Statistics and Computing, 22 (2012), pp. 773–793.

[11] M. Berveiller, B. Sudret, and M. Lemaire, *Stochastic finite element: a non intrusive approach by regression*, European Journal of Computational Mechanics, 15 (2006), pp. 81–92.

[12] B. J. Bichon, M. S. Eldred, L. P. Swiler, S. Mahadevan, and J. M. McFarland, *Efficient global reliability analysis for nonlinear implicit performance functions*, AIAA Journal, 46 (2008), pp. 2459–2468.

[13] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.

[14] G. Blatman and B. Sudret, *Sparse polynomial chaos expansions and adaptive stochastic finite elements using a regression approach*, Comptes Rendus Mécanique, 336 (2008), pp. 518–523.

[15] ———, *An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis*, Probabilistic Engineering Mechanics, 25 (2010), pp. 183–197.

[16] G. E. P. Box and N. R. Draper, *Empirical Model-Building and Response Surfaces*, John Wiley & Sons, New York, 1987.

[17] S. T. Buckland, K. P. Burnham, and N. H. Augustin, *Model selection: an integral part of inference*, Biometrics, 53 (1997), pp. 603–618.

[18] M. D. Buhmann, *Radial Basis Functions: Theory and Implementation*, Cambridge University Press, Cambridge, 2003.

[19] K. P. Burnham and D. R. Anderson, *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*, Springer, New York, 2002.

[20] D. Busby, *Hierarchical adaptive experimental design for Gaussian process emulators*, Reliability Engineering and System Safety, 94 (2009), pp. 1183–1193.

[21] F. Calio, W. Gautschi, and E. Marchetti, *On computing Gauss-Kronrod quadrature formulae*, Mathematics of Computation, 47 (1986), pp. 639–650.

[22] V. Charushnikov, *On the problem of optimizing the coefficients of cubature formulas*, Siberian Mathematical Journal, 11 (1970), pp. 714–718.

[23] H. Cheng and A. Sandu, *Collocation least-squares polynomial chaos method*, in Proceedings of the 2010 Spring Simulation Multiconference, NY, 2010, p. 1.

[24] J.-P. Chiles and P. Delfiner, *Geostatistics: Modeling Spatial Uncertainty*, John Wiley & Sons, New York, 1999.

[25] S.-K. Choi, R. Canfield, R. Grandhi, and C. Pettit, *Polynomial*

chaos expansion with latin hypercube sampling for estimating response variability, AIAA Journal, 42 (2004), pp. 1191–1198.

[26] G. Claeskens and N. L. Hjort, *Model Selection and Model Averaging*, Cambridge University Press, Cambridge, 2008.

[27] K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup, *Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients*, Computing and Visualization in Science, 14 (2011), pp. 3–15.

[28] S. D. Cohen and L. J. Guibas, *The earth mover's distance: lower bounds and invariance under translation*, tech. rep., Computer Science Department, Stanford University, 1997.

[29] T. Coleman and Y. Li, *An interior trust region approach for nonlinear minimization subject to bounds*, SIAM Journal on optimization, 6 (1996), pp. 418–445.

[30] P. R. Conrad and Y. M. Marzouk, *Adaptive Smolyak pseudospectral approximations*, SIAM Journal on Scientific Computing, 35 (2013), pp. A2643–A2670.

[31] P. R. Conrad, Y. M. Marzouk, N. Pillai, and A. Smith, *Accelerating asymptotically exact MCMC algorithms for computationally intensive models via local approximations*, Journal of the American Statistical Association (in press), (2015).

[32] P. G. Constantine, M. S. Eldred, and E. T. Phipps, *Sparse pseudospectral approximation method*, Computer Methods in Applied Mechanics and Engineering, 229-232 (2012), pp. 1–12.

[33] J. B. Conway, *A Course In Functional Analysis*, Springer, New York, 1997.

[34] R. Cools, *Advances in multidimensional integration*, Journal of Computational and Applied Mathematics, 149 (2002), pp. 1–12.

[35] S. L. Cotter, M. Dashti, and A. M. Stuart, *Approximation of Bayesian inverse problems for PDEs*, SIAM Journal on Numerical Analysis, 48 (2010), pp. 322–345.

[36] C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker, *Bayesian prediction of deterministic functions with applications to the design and analysis of computer experiments*, Journal of the American Statistical Association, 86 (1991), pp. 953–963.

[37] G. Dagan and E. Bresler, *Unsaturated flow in spatially variable*

fields: 1. Derivation of models of infiltration and redistribution, Water Resources Research, 19 (1983), pp. 413–420.

[38] P. Daniušis, D. Janzing, J. Mooij, J. Zscheischler, B. Steudel, K. Zhang, and B. Schölkopf, *Inferring deterministic causal relations*, in Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI 2010), 2010, pp. 143–150.

[39] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, Dover, Mineola, N.Y., 2007.

[40] C. R. Dietrich and G. N. Newsam, *A stability analysis of the geostatistical approach to aquifer transmissivity identification*, Stochastic Hydrology and Hydraulics, 3 (1989), pp. 293–316.

[41] P. Diggle and S. Lophaven, *Bayesian geostatistical design*, Scandinavian Journal of Statistics, 33 (2006), pp. 53–64.

[42] P. J. Diggle and P. J. Ribeiro Jr, *Model-based Geostatistics*, Springer, New York, 2007.

[43] P. J. Diggle, P. J. Ribeiro Jr, and O. F. Christensen, *An Introduction to Model-based Geostatistics*, in Spatial Statistics and Computational Methods, Springer, New York, 2003, ch. 2, pp. 43–86.

[44] A. Doostan and G. Iaccarino, *A least-squares approximation of partial differential equations with high-dimensional random inputs*, Journal of Computational Physics, 228 (2009), pp. 4332–4345.

[45] D. Draper, *Assessment and propagation of model uncertainty*, Journal of the American Statistical Association. Series B (Statistical Methodology), 57 (1995), pp. 45–97.

[46] B. Efron, *Bootstrap method: another look at the Jackknife*, Annals of Statistics, 7 (1979), pp. 1–26.

[47] M. S. Eldred, *Recent advances in non-intrusive polynomial chaos and stochastic collocation methods for uncertainty analysis and design*, in 50th AIAA/ASME/ASCE/AHS/ASC Structure, Structural Dynamics, and Materials Conference, 2009.

[48] M. S. Eldred and J. Burkardt, *Comparison of non-intrusive polynomial chaos and stochastic collocation methods for uncertainty quantification*, in 47th AIAA Aerospace Sciences Meeting, 2009, pp. 1–20.

[49] O. G. Ernst, A. Mugler, H.-J. Starkloff, and E. Ullmann, *On the convergence of generalized polynomial chaos expansions*, ESAIM: Mathematical Modelling and Numerical Analysis, 46 (2011), pp. 317–339.

[50] R. G. Gallager, *Stochastic Processes - Theory for Applications*, Cambridge University Press, Cambridge, 2013.

[51] D. Gamerman and H. F. Lopes, *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*, Chapman & Hall/CRC, Boca Raton, 2006.

[52] T. Y. Gan, E. M. Dlamini, and G. F. Biftu, *Effects of model complexity and structure, data quality, and objective functions on hydrologic modeling*, Journal of Hydrology, 192 (1997), pp. 81–103.

[53] W. Gautschi, *Orthogonal polynomials (in Matlab)*, Journal of Computational and Applied Mathematics, 178 (2005), pp. 215–234.

[54] W. Gautschi and S. Notaris, *Newton's method and Gauss-Kronrod quadrature*, Numerical Integration, 85 (1988), pp. 60–71.

[55] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*, Chapman & Hall/CRC, Boca Raton, 1995.

[56] T. Gerstner and M. Griebel, *Dimension-adaptive tensor-product quadrature*, Computing, 71 (2003), pp. 65–87.

[57] R. G. Ghanem and P. D. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Springer, New York, 1991.

[58] D. M. Ghiocel and R. G. Ghanem, *Stochastic finite-element analysis of seismic soil-structure interaction*, Journal of Engineering Mechanics, 128 (2002), pp. 66–77.

[59] M. Giles, *Multilevel Monte Carlo path simulation*, Operations Research, 56 (2008), pp. 607–617.

[60] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, Mathematics of Computation, 23 (1969), pp. 221–230.

[61] S. Haber, *Numerical evaluation of multiple integrals*, SIAM Review, 12 (1970), pp. 481–526.

[62] J. Hammersley, *Monte Carlo methods for solving multivariable problems*, Annals of the New York Academy of Sciences, 86 (1960), pp. 844–874.

[63] M. S. Handcock and M. L. Stein, *A Bayesian analysis of Kriging*, Technometrics, 35 (1993), pp. 403–410.

[64] W. K. Hasting, *Monte Carlo sampling methods using Markov chains and their applications*, Biometrika, 57 (1970), pp. 97–109.

[65] J. Hoeting, D. Madigan, A. Raftery, and C. Volinsky, *Bayesian*

model averaging: a tutorial, Statistical Science, 14 (1999), pp. 382–417.

[66] S. Hosder and R. Walters, *Efficient sampling for non-intrusive polynomial chaos applications with multiple uncertain input variables*, in 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Honolulu, Hawaii, 2007.

[67] S. Hosder, R. Walters, and R. Perez, *A non-intrusive polynomial chaos method for uncertainty propagation in CFD simulations*, in 44st AIAA Aerospace Sciences Meeting, Reno, Nevada, 2006.

[68] S. Hosder, R. W. Walters, and M. Balch, *Point-collocation non-intrusive polynomial chaos method for stochastic computational fluid dynamics*, AIAA Journal, 48 (2010), pp. 2721–2730.

[69] S. S. Isukapalli, *Uncertainty analysis of transport-transformation models*, PhD thesis, State University of New Jersey, 1999.

[70] S. S. Isukapalli, A. Roy, and P. Georgopoulos, *Stochastic response surface methods (SRSMs) for uncertainty propagation: application to environmental and biological systems*, Risk Analysis, 18 (1998), pp. 351–363.

[71] P. Jäckel, *A note on multivariate Gauss-Hermite quadrature*, Working Paper, (2005).

[72] B. Jin, *Fast Bayesian approach for parameter estimation*, International Journal for Numerical Methods in Engineering, 76 (2008), pp. 230–252.

[73] D. R. Jones, M. Schonlau, and J. William, *Efficient global optimization of expensive black-box functions*, Journal of Global Optimization, 13 (1998), pp. 455–492.

[74] G. Karagiannis and G. Lin, *Selection of polynomial chaos bases via Bayesian model uncertainty methods with applications to sparse approximation of PDEs with stochastic inputs*, Journal of Computational Physics, 259 (2014), pp. 114–134.

[75] M. C. Kennedy and A. O'Hagan, *Bayesian calibration of computer models*, Journal of the American Statistical Association. Series B (Statistical Methodology), 63 (2001), pp. 425–464.

[76] P. K. Kitanidis, *Parameter uncertainty in estimation of spatial functions: Bayesian analysis*, Water Resources Research, 22 (1986), pp. 499–507.

[77] P. K. Kitanidis, *Introduction to Geostatistics*, Cambridge University Press, Cambridge, 1997.

[78] A. Kucerová and H. G. Matthies, *Uncertainty updating in the description of heterogeneous materials*, Technische Mechanik, 30 (2010), pp. 211–226.

[79] S. Kullback and R. A. Leibler, *On information and sufficiency*, The Annals of Mathematical Statistics, 22 (1951), pp. 79–86.

[80] H. J. Kushner, *A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise*, Journal of Basic Engineering, 86 (1964), pp. 97–106.

[81] O. P. Le Maître and O. M. Knio, *Spectral Methods for Uncertainty Quantification*, Scientific Computation, Springer Netherlands, Dordrecht, 2010.

[82] O. P. Le Maître, M. T. Reagan, H. N. Najm, R. Ghanem, and O. M. Knio, *A stochastic projection method for fluid flow - II. Random process*, Journal of Computational Physics, 181 (2002), pp. 9–44.

[83] P. C. Leube, F. P. J. de Barros, W. Nowak, and R. Rajagopal, *Towards optimal allocation of computer resources: trade-offs between uncertainty quantification, discretization and model reduction*, Environmental Modelling and Software, 50 (2013), pp. 97–107.

[84] H. Li, Z. Lü, and X. Yuan, *Nataf transformation based point estimate method*, Chinese Science Bulletin, 53 (2008), pp. 2586–2592.

[85] H. Li and D. Zhang, *Probabilistic collocation method for flow in porous media: comparisons with other stochastic methods*, Water Resources Research, 43 (2007), pp. 1–13.

[86] J. Li and Y. M. Marzouk, *Adaptive construction of surrogates for the Bayesian solution of inverse problems*, SIAM Journal on Scientific Computing, 36 (2014), pp. A1163–A1186.

[87] W. Li, Z. Lu, and D. Zhang, *Stochastic analysis of unsaturated flow with probabilistic collocation method*, Water Resources Research, 45 (2009), p. W08425.

[88] J. Liesen and V. Mehrmann, *Linear Algebra*, Springer, Cham, 2015.

[89] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer, New York, 2004.

[90] P. L. Liu and A. Der Kiureghian, *Multivariate distribution models*

with prescribed marginals and covariances, Probabilistic Engineering Mechanics, 1 (1986), pp. 105–112.

[91] J. L. Loeppky, J. Sacks, and W. J. Welch, *Choosing the sample size of a computer experiment: a practical guide*, Technometrics, 51 (2009), pp. 366–376.

[92] B. P. Marchant and R. M. Lark, *Optimized sample schemes for geostatistical surveys*, Mathematical Geology, 39 (2007), pp. 113–134.

[93] J. D. Martin and T. W. Simpson, *Use of Kriging models to approximate deterministic computer models*, AIAA Journal, 43 (2005), pp. 853–863.

[94] Y. M. Marzouk and H. N. Najm, *Dimensionality reduction and polynomial chaos acceleration of Bayesian inference in inverse problems*, Journal of Computational Physics, 228 (2009), pp. 1862–1902.

[95] Y. M. Marzouk, H. N. Najm, and L. A. Rahn, *Stochastic spectral methods for efficient Bayesian solution of inverse problems*, Journal of Computational Physics, 224 (2007), pp. 560–586.

[96] Y. M. Marzouk and D. Xiu, *A stochastic collocation approach to Bayesian inference in inverse problems*, Communications in Computational Physics, 6 (2009), pp. 826–847.

[97] L. Mathelin and M. Y. Hussaini, *A stochastic collocation algorithm for uncertainty analysis*, Tech. Rep. February, NASA, 2003.

[98] T. Minka, *Deriving quadrature rules from Gaussian processes*, tech. rep., Statistics Department, Carnegie Mellon University, 2000.

[99] W. Müller, *Collecting Spatial Data*, Springer, Heidelberg, third ed., 2007.

[100] D. E. Myers, *Kriging, cokriging, radial basis functions and the role of positive definiteness*, Computers and Mathematics with Applications, 24 (1992), pp. 139–148.

[101] S. P. Neuman, *Wetting front pressure head in the infiltration model of Green and Ampt*, Water Resources Research, 12 (1976), pp. 564–566.

[102] S. P. Neuman, *Maximum likelihood Bayesian averaging of uncertain model predictions*, Stochastic Environmental Research and Risk Assessment, 17 (2003), pp. 291–305.

[103] F. Nobile, R. Tempone, and C. Webster, *A sparse grid stochastic collocation method for partial differential equations with random input data*, SIAM Journal on Numerical Analysis, 46 (2008), pp. 2309–2345.

[104] W. Nowak, *Measures of parameter uncertainty in geostatistical estimation and geostatistical optimal design*, Mathematical Geosciences, 42 (2009), pp. 199–221.

[105] W. Nowak and O. A. Cirpka, *A modified Levenberg-Marquardt algorithm for quasi-linear geostatistical inversing*, Advances in Water Resources, 27 (2004), pp. 737–750.

[106] W. Nowak, F. P. J. De Barros, and Y. Rubin, *Bayesian geostatistical design: task-driven optimal site investigation when the geostatistical model is uncertain*, Water Resources Research, 46 (2010), pp. 1–17.

[107] J. Oakley and A. O'Hagan, *Bayesian inference for the uncertainty distribution of computer model outputs*, Biometrika, 89 (2002), pp. 769–784.

[108] A. O'Hagan, *Bayes-Hermite quadrature*, Journal of Statistical Planning and Inference, 29 (1991), pp. 245–260.

[109] ———, *Bayesian analysis of computer code outputs: a tutorial*, Reliability Engineering and System Safety, 91 (2006), pp. 1290–1300.

[110] S. Oladyshkin, H. Class, R. Helmig, and W. Nowak, *A concept for data-driven uncertainty quantification and its application to carbon dioxide storage in geological formations*, Advances in Water Resources, 34 (2011), pp. 1508–1518.

[111] S. Oladyshkin, H. Class, and W. Nowak, *Bayesian updating via bootstrap filtering combined with data-driven polynomial chaos expansions: methodology and application to history matching for carbon dioxide storage in geological formations*, Computational Geosciences, 17 (2013), pp. 671–687.

[112] S. Oladyshkin and W. Nowak, *Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion*, Reliability Engineering & System Safety, 106 (2012), pp. 179–190.

[113] M. Paffrath and U. Wever, *Adapted polynomial chaos expansion for failure detection*, Journal of Computational Physics, 226 (2007), pp. 263–281.

[114] O. Pajonk, *Numerical evaluation of functionals based on variance minimisation*, Diploma Thesis, TU Braunschweig, (2008).

[115] T. Patterson, *The Optimum addition of points to quadrature formulae*, Mathematics of Computation, 22 (1968), pp. 847–856.

[116] V. Picheny, D. Ginsbourger, O. Roustant, R. Haftka, and N.-H.

Kim, *Adaptive designs of experiments for accurate approximation of target regions*, Journal of Mechanical Design, 132 (2010), p. 071008.

[117] Z.-Q. Qu, *Model Order Reduction Techniques*, Springer, London, 2004.

[118] J. Radon, *Zur mechanischen Kubatur*, Monatshefte für Mathematik, 52 (1948), pp. 286–300.

[119] A. E. Raftery, *Bayesian model selection in social research*, Sociological Methodology, 25 (1995), pp. 111–163.

[120] P. Ranjan, D. Bingham, and G. Michailidis, *Sequential experiment design for contour estimation from complex computer codes*, Technometrics, 50 (2008), pp. 527–541.

[121] M. T. Reagan, H. N. Najm, R. Ghanem, and O. M. Knio, *Uncertainty quantification in reacting-flow simulations through non-intrusive spectral projection*, Combustion and Flame, 132 (2003), pp. 545–555.

[122] T. J. Rivlin, *An Introduction to the Approximation of Functions*, Dover, Mineola, N.Y., 1969.

[123] M. Rosenblatt, *Remarks on a multivariate transformation*, The Annals of Mathematical Statistics, 23 (1952), pp. 470–472.

[124] G. Roussas, *Nonparametric Functional Estimation and Related Topics*, Springer Science+Business Media, Dordrecht, 1991.

[125] Y. Rubner, C. Tomasi, and L. J. Guibas, *A metric for distributions with applications to image databases*, Science, (1998), pp. 59–66.

[126] C. Runge, *Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten*, Zeitschrift für Mathematik und Physik, 46 (1901), pp. 224–243.

[127] D. Russo and M. Bouton, *Statistical analysis of spatial variability in unsaturated flow parameters*, Water Resources Research, 28 (1992), pp. 1911–1925.

[128] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, *Design and analysis of computer experiments*, Statistical science, 4 (1989), pp. 409–423.

[129] W. H. A. Schilders, H. A. van der Vorst, and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*, Springer, Berlin, 2008.

[130] M. Schonlau, W. J. Welch, and R. D. Jones, *Global versus local*

search in constrained optimization of computer models, New developments and applications in experimental design, 34 (1998), pp. 11–25.

[131] Y. Shin and D. Xiu, *Nonadaptive quasi-optimal points selection for least squares linear regression*, SIAM Journal on Scientific Computing, 38 (2016), pp. A385–A411.

[132] M. Sinsbeck and W. Nowak, *An optimal sampling rule for nonintrusive polynomial chaos expansions of expensive models*, International Journal for Uncertainty Quantification, 5 (2015), pp. 275–295.

[133] ———, *Sequential Design of Computer Experiments for the Solution of Bayesian Inverse Problems*, SIAM/ASA Journal on Uncertainty Quantification, 5 (2017), pp. 640–664.

[134] M. Sinsbeck and D. M. Tartakovsky, *Impact of data assimilation on cost-accuracy tradeoff in multifidelity models*, SIAM/ASA Journal on Uncertainty Quantification, 3 (2015), pp. 954–968.

[135] A. Smith and A. Gelfand, *Bayesian statistics without tears: a sampling-resampling perspective*, American statistician, 46 (1992), pp. 84–88.

[136] A. Stroud, *Approximate Calculation of Multiple Integrals*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[137] A. M. Stuart, *Inverse problems : a Bayesian perspective*, Acta Numerica, 19 (2010), pp. 451–559.

[138] B. Sudret, *Des éléments finis stochastiques spectraux aux surfaces de réponse stochastiques: une approche unifiée*, in Proceedings of the 17ème Congrès Français de Mécanique, 2005.

[139] ———, *Global sensitivity analysis using polynomial chaos expansions*, Reliability Engineering & System Safety, 93 (2008), pp. 964–979.

[140] G. R. Sun and F. F. Xiong, *Efficient sampling approaches for stochastic response surface method*, Advanced Materials Research, 538-541 (2012), pp. 2481–2487.

[141] M. H. Y. Tan, *Sequential Bayesian polynomial chaos model selection for estimation of sensitivity indices*, SIAM/ASA Journal on Uncertainty Quantification, 3 (2015), pp. 146–168.

[142] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, SIAM, Philadelphia, 2005.

[143] A. Tarantola, *Popper, Bayes and the inverse problem*, Nature Physics, 2 (2006), pp. 492–494.

[144] D. M. Tartakovsky, M. Dentz, and P. C. Lichtner, *Probability density functions for advective-reactive transport with uncertain reaction rates*, Water Resources Research, 45 (2009), p. W07414.

[145] D. M. Tartakovsky, Z. Lu, A. Guadagnini, and A. M. Tartakovsky, *Unsaturated flow in heterogeneous soils with spatially distributed uncertain hydraulic parameters*, Journal of Hydrology, 275 (2003), pp. 182–193.

[146] D. M. Tartakovsky and S. P. Neuman, *Transient effective hydraulic conductivities under slowly and rapidly varying mean gradients in bounded three-dimensional random media*, Water Resources Research, 34 (998), pp. 21–32.

[147] M. a. Tatang, W. Pan, R. G. Prinn, and G. J. McRae, *An efficient method for parametric uncertainty analysis of numerical geophysical models*, Journal of Geophysical Research, 102 (1997), pp. 21925–21932.

[148] L. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, Philadelphia, 2012.

[149] A. W. Van der Vaart, *Asymptotic Statistics*, Cambridge University Press, Cambridge, 2000.

[150] D. Venturi, D. M. Tartakovsky, A. M. Tartakovsky, and G. E. Karniadakis, *Exact PDF equations and closure approximations for advective-reactive transport*, Journal of Computational Physics, 243 (2013), pp. 323–343.

[151] J. Villadsen and M. L. Michelsen, *Solution of Differential Equation Models by Polynomial Approximation*, Prentice-Hall International, Englewood Cliffs, NJ, 1978.

[152] H. Wackernagel, *Multivariate Geostatistics: an Introduction with Applications*, Springer, Berlin, 3rd ed., 2003.

[153] W. Walker, P. Harremoës, J. Rotmans, J. van der Sluijs, M. van Asselt, P. Janssen, and M. Krayer von Krauss, *Defining uncertainty: a conceptual basis for uncertainty management in model-based decision support*, Integrated Assessment, 4 (2003), pp. 5–17.

[154] P. Wang and D. M. Tartakovsky, *Reduced complexity models for probabilistic forecasting of infiltration rates*, Advances in Water Resources, 34 (2011), pp. 375–382.

[155] A. W. Warrick, *Soil Water Dynamics*, Oxford University Press, Oxford, 2003.

[156] M. Webster, M. A. Tatang, and G. J. McRae, *Application of the probabilistic collocation method for an uncertainty analysis of a simple ocean model*, tech. rep., 1996.

[157] D. Wei, Z. Cui, and J. Chen, *Uncertainty quantification using polynomial chaos expansion with points of monomial cubature rules*, Computers & Structures, 86 (2008), pp. 2102–2108.

[158] N. A. Weiss, *A Course in Probability*, Addison-Wesley, Boston, 2005.

[159] W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris, *Screening, predicting, and computer experiments*, Technometrics, 34 (1992), pp. 15–25.

[160] N. Wiener, *The homogeneous chaos*, American Journal of Mathematics, 60 (1938), pp. 897–936.

[161] B. Williams, D. Higdon, J. Gattiker, L. Moore, M. McKay, and S. Keller-McNulty, *Combining experimental data and computer simulations, with an application to flyer plate experiments*, Bayesian Analysis, 1 (2006), pp. 765–792.

[162] B. J. Williams, T. J. Santner, and W. I. Notz, *Sequential design of computer experiments to minimize integrated response functions*, Statistica Sinica, 10 (2000), pp. 1133–1152.

[163] J. A. S. Witteveen and H. Bijl, *Modeling arbitrary uncertainties using Gram-Schmidt polynomial chaos*, in 44th AIAA Aerospace Sciences Meeting, Reno, Nevada, 2006.

[164] D. H. Wolpert and W. G. Macready, *No free lunch theorems for optimization*, IEEE Transactions on Evolutionary Computation, 1 (1997), pp. 67–82.

[165] F. Xiong, W. Chen, Y. Xiong, and S. Yang, *Weighted stochastic response surface method considering sample weights*, Structural and Multidisciplinary Optimization, 43 (2011), pp. 837–849.

[166] D. Xiu, *Efficient collocational approach for parametric uncertainty analysis*, Communications in Computational Physics, 2 (2007), pp. 293–309.

[167] D. Xiu, *Numerical Methods for Stochastic Computations - A Spectral Method Approach*, Princeton University Press, Princeton, 2010.

[168] D. Xiu and J. S. Hesthaven, *High-order collocation methods for differential equations with random inputs*, SIAM Journal on Scientific Computing, 27 (2005), pp. 1118–1139.

[169] D. Xiu and G. E. Karniadakis, *The Wiener-Askey polynomial chaos for stochastic differential equations*, SIAM Journal on Scientific Computing, 24 (2002), pp. 619–644.

[170] L. Yan, L. Guo, and D. Xiu, *Stochastic collocation algorithms using l1-minimization*, International Journal for Uncertainty Quantification, 2 (2012), pp. 279–293.

[171] E. L. Zhang, P. Feissel, and J. Antoni, *A comprehensive Bayesian approach for model updating and quantification of modeling errors*, Probabilistic Engineering Mechanics, 26 (2011), pp. 550–560.

[172] G. Zhang, D. Lu, M. Ye, M. Gunzburger, and C. G. Webster, *An adaptive sparse-grid high-order stochastic collocation method for Bayesian inference in groundwater reactive transport modeling*, Water Resources Research, 49 (2013), pp. 6871–6892.

[173] W. Zucchini, *An introduction to model selection*, Journal of Mathematical Psychology, 44 (2000), pp. 41–61.