Technische Universität München
TUM School of Computation, Information and Technology

Optimization under uncertainty and the multilevel Monte Carlo method

Friedrich M. Menhorn

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 23.08.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 07.03.2024 angenommen.

Acknowledgements

This was always the part I was most looking forward to writing. Thanks is long overdue to a lot of people involved in this work in various ways. First of all, I would like to express my gratitude to my supervisor, Prof. Hans-Joachim Bungartz. Thank you for letting me pursue my research freely and letting me have my own experiences. Whenever guidance was needed, your door was always open and I could count on your advice. Thank you also for letting me follow other adventures though they might sometimes interfere with this PhD. I want to also say thank you to my second advisor, Prof. Youssef Marzouk, who directed me on this path of (optimization under) uncertainty when offering me the chance for a master thesis. This was a life-changing experience. Thank you for also letting me visit you during my PhD and supporting me with advice and support throughout this work. Thanks to the both of you for reviewing this thesis. I also want to say thank you to Prof. Nils Thurey for chairing my PhD committee.

Thank you to my mentor, Dr. Dr. Matthias Reumann, who was possibly the first person giving me the idea of starting a PhD and supported me throughout this career with his experience and advice. Thank you to Dr. Florian Augustin, who supported me during my first visit to MIT and continuously gave me feedback even though he unfortunately left the world of academia. Very importantly, I would like to thank Dr. Tobias Neckel, who brought me to the chair of Scientific Computing, introduced me to Uncertainty Quantification and, most likely, supported me the longest. We collaborated on so many projects apart from the PhD that I am not able to count them. Thank you for our great collaboration and your patience all these years. Next, I would like to thank all my colleagues at the chair of Scientific Computing, specifically the Uncertainty Quantification family with Ivana and Kislaya and my study and work colleagues and friends Severin, Benjamin, Obi and Michael. In particular, I want to say thank you to Ionuț, a long-term colleague and friend, now-turned advisor and proof-reader of this work. I would like to thank my students, who contributed to this work in alphabetical order: Marina Baumgartner, Jonas Donhauser, Amr Elsharkawy and Kislaya Ravi.

I would also like to express my gratitude to collaborators and contributors to this work. From Sandia National Laboratories, thank you to Mike Eldred and Gianluca Geraci for the great collaboration over the last years and a special thanks for letting me stay and work with you in Albuquerque. Thank you for this great experience. Moreover, thank you to Tom Seidl for his constant and meticulous support and the Dakota family (Brian Adams, J. Adam Stephens and Russell Hooper) for letting me contribute to their software. It was great to feel a part of and get insight into your work. Thank you to Ryan King from the National Renewable Energy Laboratory for his help and support with the wind application. This interdisciplinary thesis would definitely not have been possible without the contributions and help from all of you.

Of course, this work is not doable without support from persons in my private life. I want to thank my family, my parents and sisters for always supporting me and having my back. I want to specifically thank my sister for proofreading this work. All the mistakes you cannot find are not there because of her. I want to thank my girlfriend, who encouraged me, aided me and reduced the struggles of daily life, especially during the last months leading to the completion of this thesis.

Last but by far not least, I have to thank you again, Gianluca. Your guidance, advice, realism and patience played a major role in the completion of this work. I am unbelievably grateful for your commitment in time, energy and nerves, and thank you specifically once more.

Abstract

Optimization plays a key role in scientific and engineering fields where the objective is to find optimal designs and solutions. To enhance realism, it is necessary to consider uncertainty in the optimization problem. As a result, the optimization formulation must be extended to encompass statistics of the quantity of interest. Ultimately, the goal is to identify a solution that is robust and reliable, taking into account probabilistic constraints. This is where Optimization under Uncertainty (OUU) comes into play.

However, OUU is computationally very challenging. Stochastic parameters and calculating statistics significantly increase computational demands. Instead of solving a deterministic problem only once, we must solve it multiple times to compute them. This is aggravated in OUU, where we must solve statistics at each optimization step.

In general, an OUU workflow involves two primary components: an inner-loop sampling strategy to compute statistics, and an outer-loop optimization strategy to identify the optimal design based on the merit function derived from those statistics. For both components, this thesis presents major improvements to handle the increased computational demands.

For the inner-loop component, the multilevel Monte Carlo (MLMC) method is able to alleviate the cost of the uncertainty analysis considerably by distributing resources among multiple models with varying accuracy and cost. While previous approaches focused on the expected value, we devise novel MLMC estimators for the variance, standard deviation, and a linear combination of expected value and standard deviation. Developing these estimators is crucial, since these statistics are frequently utilized in OUU workflows and can increase the computational performance by order of magnitudes.

For the outer-loop component, we propose the derivative-free optimization method SNOWPAC. SNOWPAC employs a trust-region approach using fully-linear surrogates combined with Gaussian Process surrogates. Compared to many other derivative-free approaches, SNOWPAC is able to handle stochastic constraints. Furthermore, we present extensions of the method using approximate Gaussian Processes to alleviate the computational cost of the surrogate, adaptive kernel selection to improve performance of the surrogate and an extension of the method for mixed-integer optimization problems.

We couple the new MLMC approach with the new optimization method SNOWPAC in the software toolkit Dakota to address complex black-box problems in OUU. We verify the performance of our proposed method using various benchmarks, including an illustrative one-dimensional problem, the two-dimensional Rosenbrock function, and a benchmark test suite based on the CUTEst benchmark set. Finally, we utilize our novel approach to optimize a wind power plant, where our objective is to determine the optimal yaw alignment of the turbines to maximize power production.

Zusammenfassung

Die Optimierung spielt eine Schlüsselrolle in wissenschaftlichen und technischen Bereichen, in denen es darum geht, optimale Lösungen zu finden. Um die Realität besser abzubilden zu können, ist es notwendig, Unsicherheiten im Optimierungsproblem zu berücksichtigen. Daher muss die Optimierungsformulierung erweitert werden, um auch Statistiken der interessierenden Größe einzubeziehen. Letztlich geht es darum, eine robuste und zuverlässige Lösung zu finden, die probabilistische Beschränkungen berücksichtigt. Hier kommt die Optimierung unter Unsicherheit (OUU) ins Spiel.

Allerdings ist die OUU rechnerisch sehr anspruchsvoll. Stochastische Parameter und die Berechnung von Statistiken erhöhen die Rechenanforderungen erheblich. Ein deterministisches Problem muss nun vielfach gelöst werden, um diese Statistiken zu berechnen. Dies wird bei OUU noch weiter verschärft, wo in jedem Optimierungsschritt Statistiken berechnet werden müssen.

Im Allgemeinen umfasst ein OUU-Arbeitsablauf zwei Hauptkomponenten: eine Strategie für die innere Schleife zur Berechnung von Statistiken, und eine Optimierungsstrategie für die äußere Schleife, zur Ermittlung des optimalen Designs auf der Grundlage der aus diesen Statistiken abgeleiteten Zielfunktion. Für beide Komponenten werden in dieser Arbeit wesentliche Verbesserungen vorgestellt, um die erhöhten Rechenanforderungen zu bewältigen.

Für die Inner-Loop-Komponente kann die multilevel Monte-Carlo-Methode (MLMC) die Kosten der Unsicherheitsanalyse erheblich verringern, indem sie die Ressourcen auf mehrere Modelle mit unterschiedlicher Genauigkeit und Kosten verteilt. Während sich frühere Ansätze auf den Erwartungswert konzentrierten, entwickeln wir neue MLMC-Schätzer für die Varianz, die Standardabweichung, sowie eine lineare Kombination aus Erwartungswert und Standardabweichung. Die Entwicklung solcher Schätzer ist von entscheidender Bedeutung, da diese Statistiken häufig in OUU-Arbeitsabläufen verwendet werden und die Rechenleistung um mehrere Größenordnungen steigern können.

Für die Komponente der äußeren Schleife präsentieren wir die ableitungsfreie Optimierungsmethode SNOWPAC. SNOWPAC verwendet einen Trust-Region-Ansatz, der vollständig lineare Surrogate mit Gauß-Prozess-Surrogaten kombiniert. SNOWPAC ist in der Lage stochastische Nebenbedingungen zu behandeln, im Vergleich zu vielen anderen ableitungsfreien Methoden. Darüber hinaus stellen wir Erweiterungen der Methode vor, die approximative Gauß'sche Prozesse verwenden, um die Rechenkosten des Surrogats zu verringern, eine adaptive Kernelauswahl, um die Leistung des Surrogats zu verbessern, und eine Erweiterung der Methode für gemischt-ganzzahlige Optimierungsprobleme.

Wir koppeln den neuen MLMC-Ansatz mit der neuen Optimierungsmethode SNOWPAC im Software-Toolkit Dakota, um komplexe Black-Box-Probleme in OUU zu lösen.

Wir verifizieren die Leistungsfähigkeit der von uns vorgeschlagenen Methode anhand verschiedener Benchmarks, darunter ein illustratives eindimensionales Problem, die zweidimensionale Rosenbrock-Funktion und eine Benchmark-Testsuite, die auf dem CUTEst-Benchmark-Set basiert. Schließlich verwenden wir unseren Ansatz zur Optimierung einer Windkraftanlage, wobei unser Ziel darin besteht, die optimale Ausrichtung der Turbinen zu bestimmen, um die Stromproduktion zu maximieren.

Contents

Part I

Introduction

As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality.

—Albert Einstein [109]

1 Uncertainty quantification pipelines

Numerical modeling captures the laws of physics through models of the real world. Numerical simulations simulate such models on a computer with the intention to be able to comprehend and predict phenomena of the real world. The underlying motivation behind is manifold: one is certainly to reduce monetary costs: we could crash a car against a wall in real life or simulate it many times over on a machine. Another is being able to predict reality: through weather and climate simulations, we are able to forecast whether there will be sunshine in the north east of Munich tomorrow at 3 pm, or if we need to bring an umbrella. One further motivation is to get insight: numerical simulation allows us to simulate processes that are otherwise not yet realizable or feasible. We cannot travel into a black hole or into the sun to analyse events happening inside. Via simulation, we get an understanding of the very small, such as the quantum world; or the very large, such as plasma fusion processes in stars. What is more, we get this insight without burning down the full facility (and the land surrounding it).

The main ingredients of a modeling and simulation pipeline, visualized in Fig. 1.1, are threefold: in the center we have the model of reality. It could be an ordinary differential equation (ODE), such as a population model to help us predict the population of two species in a predator-prey model. It could be derived from partial differential equation (PDE) that models a natural phenomenon such as the Navier-Stokes equations for the simulation of fluids. Also, other models such as fuzzy logic or a consortium of different models that are linked together—typical for climate models—can be put here. On the left, we have the input parameters, e.g. the starting population for evolution models, the inflow velocity or pressure for the Navier-Stokes equations or measurements from sensors all over the world for climate simulations. As output of the model, we get our quantity of interest (QoI), depicted on the right. This could either be a single number such as the power production of a wind turbine in the field, or a collection of numbers such as the temperature distribution all over Germany.

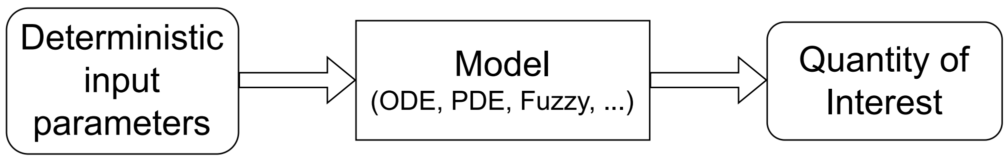


Figure 1.1: Deterministic simulation pipeline.

Until now, we only considered deterministic scenarios. While such models and their simulation can be very accurate, they still simplify reality. To improve the realism of our simulations further, we include uncertainty: we view uncertainty as additional information—paradoxically to be fair—that we can use to gain more insight. Already knowing that a parameter is not necessarily deterministic, but can vary in a certain range, allows us to better predict its behavior. When we know that our model introduces numerical errors or uses approximations of reality, we can use that knowledge to increase (or at least estimate) the confidence in our prediction. Nowadays, we also have access to immense amounts of data we can use to improve our simulations. Measuring not only data but also assessing the error of the corresponding sensor helps us in finding more realistic solutions.

In the field of *uncertainty quantification* (UQ), we treat and incorporate these uncertainties to help us model reality; a reality which is stochastic by nature. We consider *forward uncertainty propagation* (or *forward UQ*) in this thesis. Forward uncertainty propagation propagates, as the name suggests, the uncertainty of input parameters forward through a model, and estimates QoIs which are now, as a result, of stochastic nature. We have outlined the pipeline for forward UQ in Fig. 1.2.

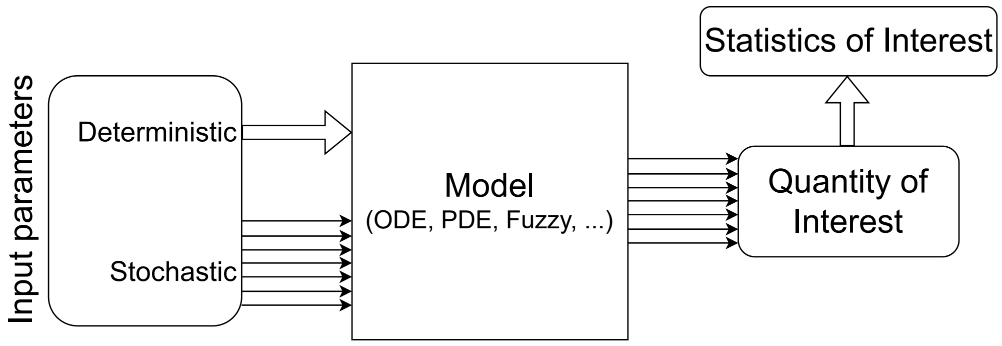


Figure 1.2: Forward uncertainty quantification pipeline.

We now not only consider deterministic but also stochastic parameters as input. While the deterministic parameters are fixed, the uncertain inputs are generally modelled as random variables whose probability distribution typically stems from experiments, expert opinion or a combination thereof. Both the deterministic as well as the stochastic parameters are propagated through the model. As a result, instead of deterministic values, we receive a set of QoI from which we deduce statistics of interest (SoI). Common statistics are the expected value (or mean) or the variance.

Computationally, we should be able to sample the stochastic parameter space to get samples (or realizations). Each of the samples, together with the deterministic variables, is put through the model, resulting in a set of QoI for given deterministic parameters. Finally, from this set, we can compute the SoI. Since we have to evaluate the model now multiple times for the set of stochastic realizations to compute the statistics, the solution naturally grows in computational complexity and expense. Hence, we have to find strategies to lower the computational burden while still realizing high accuracy.

2 Model hierarchies

A way to lower computational cost is employing a *model hierarchy*. Standard simulation practice involves a single model, which solves the input-to-output relation in simulation pipelines such as Fig. 1.1 or Fig. 1.2. This could be a *high-fidelity* model, meaning a complex and high-dimensional model that has very high accuracy accompanied by high computational cost; or, if the cost of developing and utilizing such models is prohibitively high, it could be a *low-fidelity* model. A low-fidelity model is less accurate due to, e.g., dimensionality reduction, linearization, use of simpler physics models, coarser domains, or partially converged results. In these cases, we often speak of *reduced* or *surrogate* models. These are models, which approximate the high-fidelity model, but trade-off accuracy for smaller computational cost.

However, we often have multiple models available that model the same (or at least similar) physical phenomenon. All models add different information and, in the best case, we can combine them to enrich our simulation. We can order such sets of models in a model hierarchy: at the lower end of the hierarchy, we have models that are computationally cheap but inaccurate. Climbing up the hierarchy, the models get more and more accurate at the price of increased computational cost. We differentiate between two main model hierarchies: *fidelities* and *levels*.

We speak of different fidelities if models describe the same problem but use different approaches or modeling choices at various levels of accuracy and computational cost. A common example is computational fluid dynamics, where incompressible Navier-Stokes represent a lower fidelity, while computational cost and accuracy grow with fidelities such as Reynolds-Averaged Navier-Stokes, Large Eddy Simulation or Direct Navier-Stokes. When developing methods for such problems, we speak of *multi-fidelity methods*.

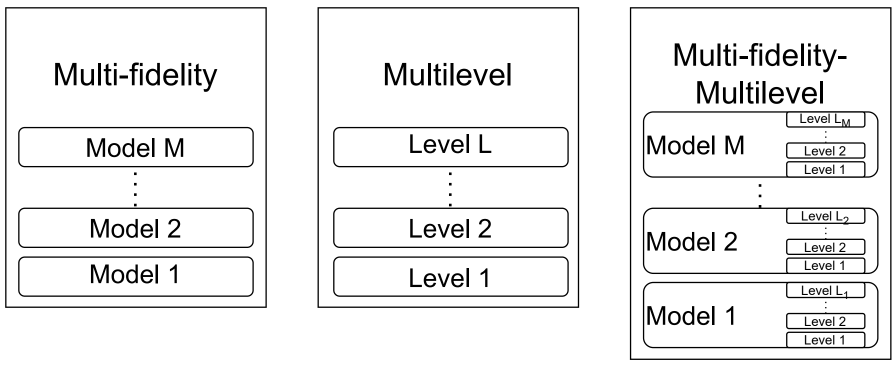


Figure 2.1: Outline of multi-fidelity (left), multilevel (center) and multi-fidelity-multilevel hierarchies (right).

A level hierarchy is typically obtained by varying parameters characterizing the high-fidelity model such as the computational mesh width or the variances of its input parameters. Repeating the example for Navier-Stokes, we could utilize incompressible Navier-Stokes equations across different spatial grid discretizations. A coarser grid incurs lower computational cost but at the same time offers lower accuracy, whereas a finer grid, although computationally more expensive, yields more precise results. We call methods of this type *multilevel methods*. Both approaches can also be mixed, by designing a multilevel hierarchy for each fidelity, resulting in *multi-fidelity-multilevel methods*. We outline the different hierarchies in Fig. 2.1.

While a multi-fidelity or multilevel hierarchy can result in large computational benefits (or highly improved accuracy), it also comes with multiple challenges. The question of how to pick and order the hierarchy is often complex. This requires a significant understanding of the problem at hand and the relationship between various aspects of the model. It can be challenging to determine the appropriate balance between accuracy and computational cost for each level of the model hierarchy. Additionally, a cost reduction is not guaranteed by default. The cost of creating and maintaining the low-fidelity models, transferring information between models, and checking for errors may outweigh the potential savings. Finally, when working with existing, or legacy, computational code, it can be challenging to implement multilevel and multi-fidelity methods without significant code modifications or redesign. This can lead to increased development time and potential errors. Nevertheless, the significant benefits of properly designed hierarchies outweigh these challenges.

3 Outer-loop formulations in uncertainty quantification

Until now, we have only been looking at an input-output relationship in the UQ pipeline, where we know the input, propagate it through a model (or model hierarchy) and are interested in the output. However, it might also be the case that we know output and model, but not the stochastic input, or we know input and output but are interested in the model. In these cases, we iteratively try to adapt our input data or model to the given output. Such problems are called an *outer-loop* formulation. We still have the inner pipeline, but now we have an outer loop which updates are based on an error or loss.

An example of an outer-loop formulation are *inverse problems*, where in contrast to forward UQ, we have data of a QoI and need an estimate of the input parameters. If this estimate is of stochastic nature, i.e. we try to estimate a distribution of the stochastic input parameters based on data, we call this *Bayesian inference* or *backward inference*. We illustrate the loop in Fig. 3.1. The data allows us to compute an error with respect to our prediction to update the input. The estimate of the input is then iteratively updated. In this work, we focus on forward propagation of uncertainty and only mention this branch as an extension and for completeness.

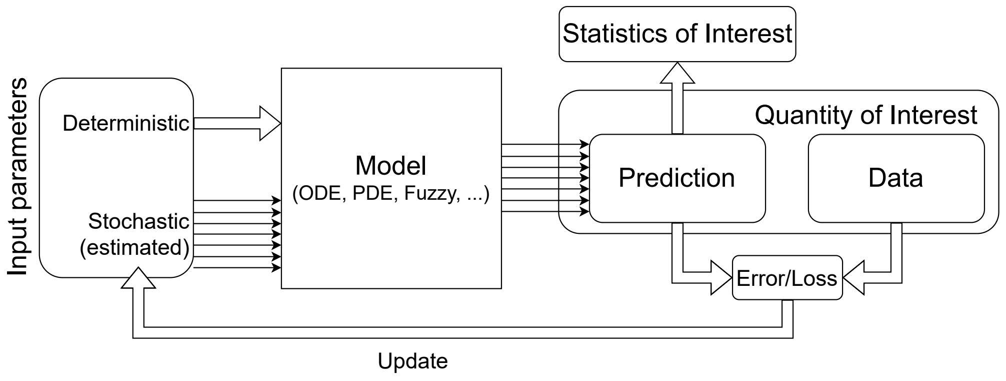


Figure 3.1: Bayesian inference loop.

Another very important area in simulation and modeling, which has seen exponentially increased interest over the last decade and another example of outer-loop formulation, is machine learning. We visualize the loop in Fig. 3.2. Here, we are not necessarily interested in output statistics as in forward propagation. We moreover already know the input parameters in contrast to inverse problems. Instead, we want to find the optimal model, where we have input and output data available[1]. In this case, the model is iteratively updated based on the data. Note that this data is also usually stochastic in nature, e.g., due to measurement errors in the sensors or different measurement conditions. After training the model using the data, we are able to use it for predictions for new inputs. Neural networks and Gaussian Processes are popular methods in this field. We are going to discuss both topics in this work.

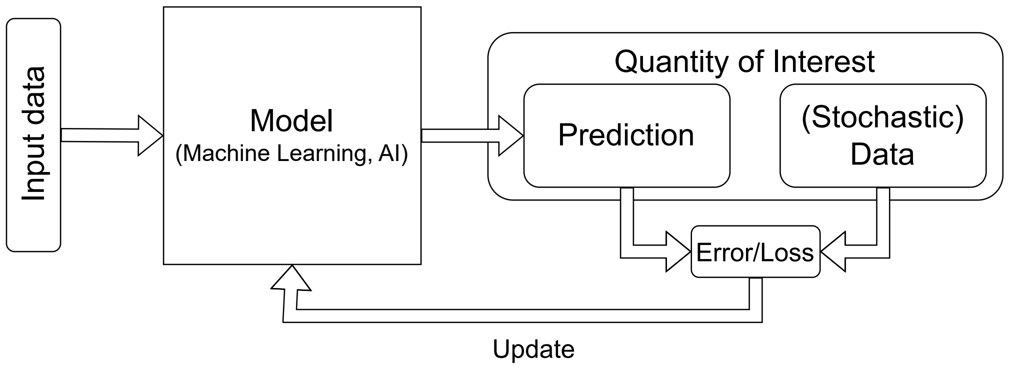


Figure 3.2: Machine learning loop for supervised learning.

[1]This considers supervised learning only for simplification.

4 Optimization under uncertainty

We have looked at outer-loop formulations where we are interested in the stochastic input parameters in Bayesian inference, or the model in machine learning. However, we can also ask the following question: What is the optimal choice of deterministic inputs to yield the best value for a chosen output statistic, given the inherent stochastic parameters?

This leads us to an optimization problem: in optimization, we are interested in finding the optimal *design* to minimize (or maximize) a given *objective*, possibly under specific *constraints*. Assuming a realistic and complex model, this optimization process proceeds iteratively, updating the current best design and objective until convergence. In the deterministic setting, which we visualize in Fig. 4.1, this involves repeatedly computing the inner simulation pipeline from Fig. 1.1. However: what if our forward model involves uncertain parameters to improve the realism of our simulations further?

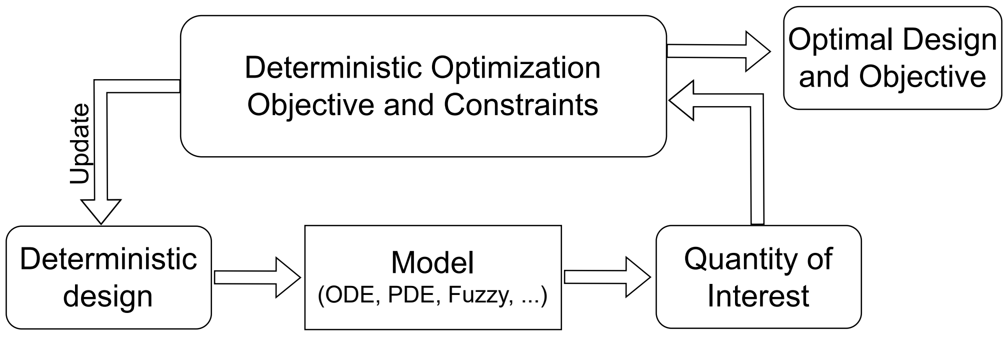

Deterministic Optimization
Objective and Constraints
Optimal Design and Objective
Update
Deterministic design
Model
(ODE, PDE, Fuzzy, ...)
Quantity of Interest

Figure 4.1: Deterministic optimization loop.

Revisiting the Navier-Stokes example, we may want to find the optimal design of the airfoil of the new Boeing 777X in the wind channel under these uncertain conditions. Or, as we will discuss in this work, we aim to determine the best yaw angle setting for a group of wind turbines deployed off the coast of Norway in order to maximize the average power production of that wind power plant. In this case, we talk about *optimization under uncertainty* (OUU) where we try to find an optimal solution with respect to those uncertainties. We can try to, e.g., maximize the average outcome or try to limit the variation of the QoI.

We still have the forward uncertainty propagation as the inner part, but now have an outer optimization loop on the deterministic input, which—given the current best value of the *objective* under some *constraints*—picks the next iteration of a deterministic *design* based on some optimization rule under stochastic conditions. We visualize the pipeline in Fig. 4.2.

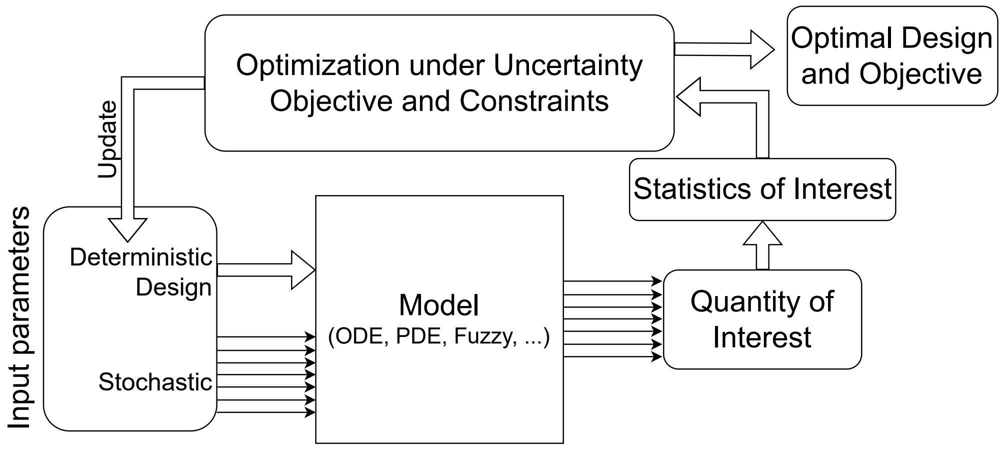


Figure 4.2: Optimization under uncertainty loop.

The outer-loop optimization adds another dimension to the computational cost: the cost stemming from the deterministic case are already multiplied by the number of stochastic samples needed for forward UQ. This is now further aggravated by the optimization loop, which iteratively calls upon the inner loop until an optimal solution is found. While this can significantly increase computational cost, it simultaneously provides multiple angles and opportunities where cost optimization is possible.

In total, we thus have three layers where we can improve computational performance: on the top layer is the optimization algorithm where we can work on fast convergence to require the least amount of optimization steps possible. For the forward UQ problem, on the mid layer, we can fall back to efficient methods in that field, such as multilevel methods, to speed up computations of statistics. Finally, on the innermost layer—the model itself—we can improve performance of the algorithms, e.g., the involved linear algebra or the integration in time in the solution of PDEs. Hence, as we will see in this thesis, a lot of different research fields can contribute to different layers to tackle challenging problems in OUU.

5 Preview

In this thesis, we concentrate on the two outer layers of OUU to optimize its performance and present three contributions:

Our first major contribution focuses on the mid layer. We provide new multilevel estimators for higher-order moments, namely the variance, standard deviation, and a linear combination of mean and standard deviation. These estimators prove vital in the multilevel setting, especially when the OUU formulation involves the said statistics. The developments around these estimators are presented in Part III, accompanied by a benchmark example. With these new estimators, we can lower the computational cost of the second layer, also specifically for their application in OUU.

Our second contribution improves the outer layer by introducing a novel stochastic optimization algorithm, named SNOWPAC, intended for OUU. Employing a derivative-free approach, this algorithm leverages trust regions and treats the model as a black box. Part IV presents SNOWPAC, along with key enhancements and improvements made to the algorithm over recent years. SNOWPAC provides an efficient stochastic optimizer that converges quickly despite the underlying uncertainty and thus reduces the computational cost of the outermost layer.

Finally, in the third contribution, we bring all contributions together in a practical context, as we tackle OUU problems in Part V, implemeting the full pipieline of Fig. 4.2. We regard a one-dimensional benchmark scenario and the constrained Rosenbrock function. For both benchmarks, we design a new multilevel hierarchy where we can compare our contribution against a reference solution. To conclude, we regard a realistic wake-steering optimization problem for wind power plants: we design a multilevel case with three levels and aim to find the optimal selection of yaw angles for all involved turbines under stochastic conditions, demonstrating the practical application and utility of our work.

However, before we dive into our contributions, we begin with the background of this work in Part II: We detail the basis of forward UQ in general and sampling in particular in Chap. 6. This chapter also includes a specific focus on the Monte Carlo method. Next, we discuss Gaussian Process surrogates in Chap. 7, an essential component in our work, especially for Part IV. Afterwards, we talk about derivative-free optimization in Chap. 8 and derivative-free optimization under uncertainty specifically in Chap. 9. Both include a targeted focus on trust-region methods. We present the software framework Dakota in Chap. 10 before we end the background part with a gap analysis in Chap. 11. All background chapter include a thorough review of literature in the field.

Part II

Theoretical background

We demand rigidly defined areas of doubt and uncertainty!

—*Douglas Adams [5]*

6 Uncertainty propagation and sampling methods

In the first part of the theoretical background, we cover the forward UQ pipeline introduced in Chap. 1, which represents the second layer in an outer-loop formulation in OUU. Methods in forward UQ lie at the base of our work to efficiently propagate uncertainty through a model and compute SoI. We first discuss the UQ setting in Section 6.1 followed by a review of methods in forward UQ in Section 6.2. In Section 6.3, we focus on Monte Carlo sampling and finalize this chapter by introducing multilevel Monte Carlo in Section 6.4, which is a main component of our first contribution in Part III.

6.1 Random variables, quantities and statistics of interest

We begin by formalizing the notation and problem that we want to solve: We denote $d_{\text{det}} \in \mathbb{N}$ as the dimension of our deterministic space, while $d_{\text{sto}} \in \mathbb{N}$ represents the stochastic dimensionality. We introduce $\theta : \Omega \to \mathbb{R}^{d_{\text{sto}}}$ as a *multivariate real-valued random variable*, which is defined in the complete probability space $(\Omega, \mathcal{F}, P)$. The variable Ω is a non-empty set representing the sample space; $\mathcal{F} \subset 2^{\Omega}$ is a σ-algebra of events; the probability measure, $P : \mathcal{F} \to [0, 1]$, is a non-negative measure and assigns each event in the event space a probability between 0 and 1. We further use the simplified notation $\theta := \theta(\boldsymbol{\omega})$ or $\theta_i := \theta(\omega_i)$ for each component, $\{\omega_i\}_{i=1}^{d_{\text{sto}}}$.

Moreover, θ is assumed to be a *continuous* random variable which is characterized by a *probability density function* $p : \Omega \to [0, 1]^{d_{\text{sto}}}$(or $p_i : \Omega_i \to [0, 1]$ for each component). Finally, we assume that all $\{\theta_i\}_{i=1}^{d_{\text{sto}}}$ are L^2 random variables, i.e. they have a finite expectation $< \infty$,

$$\mathbb{E}[\theta_i] = \int_{\Omega_i} \theta_i p_i(\theta_i) d\theta_i \tag{6.1}$$

and finite variance $< \infty$

$$\mathbb{V}[\theta_i] = \mathbb{E}[\theta_i^2] - \mathbb{E}[\theta_i]^2. \tag{6.2}$$

We specify the numerical model that represent the complex real-world phenomena as $Q : \mathbb{R}^{d_{\text{det}}} \times \mathbb{R}^{d_{\text{sto}}} \to \mathbb{R}$. Here, $\mathbf{x} := [x_1, x_2, ..., x_{d_{\text{det}}}]^T \in \mathbb{R}^{d_{\text{det}}}$ and $\boldsymbol{\theta} := [\theta_1, \theta_2, ..., \theta_{d_{\text{sto}}}]^T \in \mathbb{R}^{d_{\text{sto}}}$ denote the vector notations of both the deterministic and the stochastic input vectors. We generally employ a bold notation for vector variables throughout this work.

The model Q is a mapping from a d_{det}-dimensional deterministic input, $\mathbf{x}$, and d_{sto}-dimensional stochastic input, $\boldsymbol{\theta}$, to a scalar value[1]— our *quantity of interest* (QoI). Note

[1]Note that we could also map to a real vector as output, but we simplify notation without loss of generality to a one-dimensional output.

that we make a simplification here, denoting the model directly as mapping from the input to the QoI (instead of using a second mapping from model output to QoI). We also point out that Q is already a high-fidelity, discretized, numerical solution and we neglect the discretization error throughout this work. When possible, we also simplify notation using $Q := Q(\mathbf{x}, \boldsymbol{\theta})$ for brevity, referring to Q as our QoI directly.

In the end, we are interested in *statistics of interest* (SoI) of our QoI. We specifically look at two statistics:

- expected value:
$$\mathbb{E}[Q(\mathbf{x}, \boldsymbol{\theta})] := \int_\Omega Q(\mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{6.3}$$
- variance:
$$\mathbb{V}[Q(\mathbf{x}, \boldsymbol{\theta})] := \int_\Omega (Q(\mathbf{x}, \boldsymbol{\theta}) - \mathbb{E}[Q(\mathbf{x}, \boldsymbol{\theta})])^2 p(\boldsymbol{\theta}) d\boldsymbol{\theta} = \mathbb{E}[Q(\mathbf{x}, \boldsymbol{\theta})^2] - \mathbb{E}[Q(\mathbf{x}, \boldsymbol{\theta})]^2, \tag{6.4}$$

and one derived statistic:

- standard deviation:
$$\sigma[Q(\mathbf{x}, \boldsymbol{\theta})] := \sqrt{\mathbb{V}[Q(\mathbf{x}, \boldsymbol{\theta})]}. \tag{6.5}$$

Apart from these three main SoI, we also define the i-th central moment statistic, which we need occasionally throughout this work, as

$$\mu_i[Q(\mathbf{x}, \boldsymbol{\theta})] := \int_\Omega (Q(\mathbf{x}, \boldsymbol{\theta}) - \mathbb{E}[Q(\mathbf{x}, \boldsymbol{\theta})])^i p(\boldsymbol{\theta}) d\boldsymbol{\theta} \text{ for } i > 1. \tag{6.6}$$

Note that this means that $\mu_2[Q(\mathbf{x}, \boldsymbol{\theta})] = \mathbb{V}[Q(\mathbf{x}, \boldsymbol{\theta})]$. We define $\mu_1[Q(\mathbf{x}, \boldsymbol{\theta})] := \mathbb{E}[Q(\mathbf{x}, \boldsymbol{\theta})]$ as an exception.

Computing statistics such as the mean, variance or standard deviation requires evaluating multivariate integrals, in our case of dimension d_{sto}. The challenge here is that the integrand, which depends on the underlying numerical model, is very rarely available in closed form. These potentially high-dimensional integrals must be evaluated numerically as a consequence. This, in turn, requires discrete realizations of $\boldsymbol{\theta}$. We denote these discrete realizations as *samples*, $\boldsymbol{\theta}_i = \theta(\boldsymbol{\omega}_i)$, from a specific event $\boldsymbol{\omega}_i$, which in turn result in a sample of our QoI $Q^{(i)} := Q(\mathbf{x}, \boldsymbol{\theta}_i)$ after solving the forward model. Those samples can be a result of (pseudo-)random numbers, but can also be deterministically chosen, as a result from, e.g., collocation or quadrature points to solve the integral. Due to the discrete nature, the quadrature is only an approximation of the integral solution. Hence, we use $\widehat{\mu_i} \approx \mu_i, i \geq 1$ as our symbol for the approximation of the SoI. We call $\widehat{\mu_i}$ *estimator* of the statistic μ_i, and mention again specifically the estimators for mean $\widehat{\mu}_1[Q(\mathbf{x}, \boldsymbol{\theta})] \approx \mathbb{E}[Q(\mathbf{x}, \boldsymbol{\theta})]$, variance $\widehat{\mu}_2[Q(\mathbf{x}, \boldsymbol{\theta})] \approx \mathbb{V}[Q(\mathbf{x}, \boldsymbol{\theta})]$ and standard deviation $\widehat{\sigma}[Q(\mathbf{x}, \boldsymbol{\theta})] \approx \sigma[Q(\mathbf{x}, \boldsymbol{\theta})]$.

It is important to note that an estimator of a statistic is itself a random variable. Given a statistic X and a generic estimator $\hat{X}$, we define two important properties of estimators that we use throughout this work:

The first property is the *bias* of an estimator

$$\mathrm{BIAS}(\hat{X}) = \mathbb{E}[\hat{X}] - X. \tag{6.7}$$

We call an estimator *unbiased*, if $\mathrm{BIAS}(\hat{X}) = 0$, i.e. $\mathbb{E}[\hat{X}] = X$. If an estimator is biased, we denote this explicitly as subscript $\hat{X}_{\text{biased}}$.

The second property is the *root mean squared error* (*RMSE*), which holds information about the approximation quality of the estimator:

$$\mathrm{RMSE}(\hat{X}) = \sqrt{\mathbb{E}[(\hat{X} - X)^2]}. \tag{6.8}$$

We can rewrite the RMSE in a different form. This is a well-known result, but due to its importance in this work we state in form of the following lemma:

Lemma 1. *Given a statistic X and an estimator $\hat{X}$ such that $\hat{X} \approx X$, the RMSE is given as*

$$\mathit{RMSE}(\hat{X}) = \sqrt{\mathit{BIAS}(\hat{X})^2 + \mathbb{V}[\hat{X}]}. \tag{6.9}$$

Proof.

$$\begin{aligned}
\mathrm{RMSE}(\hat{X}) &= \sqrt{\mathbb{E}[(\hat{X} - X)^2]} \\
&= \sqrt{\mathbb{E}[\hat{X}^2 - 2\hat{X}X + X^2]} \\
&= \sqrt{\mathbb{E}[\hat{X}^2] - 2\mathbb{E}[\hat{X}X] + \mathbb{E}[X^2]} \\
&= \sqrt{\mathbb{E}[\hat{X}^2] - 2\mathbb{E}[\hat{X}]X + X^2} \\
&= \sqrt{\mathbb{E}[\hat{X}^2] - 2\mathbb{E}[\hat{X}]X + (\mathbb{E}[\hat{X}] - \mathrm{BIAS}(\hat{X}))^2} \\
&= \sqrt{\mathbb{E}[\hat{X}^2] - 2\mathbb{E}[\hat{X}]X + \mathbb{E}[\hat{X}]^2 - 2\mathbb{E}[\hat{X}]\mathrm{BIAS}(\hat{X}) + \mathrm{BIAS}(\hat{X})^2} \\
&= \sqrt{\mathbb{E}[\hat{X}^2] - 2\mathbb{E}[\hat{X}]X + \mathbb{E}[\hat{X}]^2 - 2\mathbb{E}[\hat{X}](\mathbb{E}[\hat{X}] - X) + \mathrm{BIAS}(\hat{X})^2} \\
&= \sqrt{\mathbb{E}[\hat{X}^2] - 2\mathbb{E}[\hat{X}]X + \mathbb{E}[\hat{X}]^2 - 2\mathbb{E}[\hat{X}]\mathbb{E}[\hat{X}] + 2\mathbb{E}[\hat{X}]X + \mathrm{BIAS}(\hat{X})^2} \\
&= \sqrt{\mathbb{E}[\hat{X}^2] - \mathbb{E}[\hat{X}]^2 + \mathrm{BIAS}(\hat{X})^2} \\
&= \sqrt{\mathbb{V}[\hat{X}] + \mathrm{BIAS}(\hat{X})^2}
\end{aligned} \tag{6.10}$$

□

Based on this property, a main focus (and challenge) of this thesis lies in finding unbiased estimators. Given the unbiased estimators, we can compute the RMSE using the variance, i.e. $\mathrm{RMSE}(\hat{X}) = \mathbb{V}[\hat{X}]$ and ignore the bias term.

In the following section, we summarize some of the existing, prominent methods for forward UQ. Afterwards, we focus on Monte Carlo sampling, which is of interest for the remainder of this thesis.

6.2 Stochastic collocation and the Galerkin method

In general, we distinguish between three classes of methods for forward uncertainty propagation [116]—(*intrusive*) *white-box*, (*non-intrusive*) *grey-box* and (*non-intrusive*) *black-box* methods. The naming scheme is referencing access to the underlying model and the difference between the three classes is already hinted at in their name. Let us elaborate this difference further.

In intrusive methods, we need access to the underlying problem and be able to modify it. Practically speaking, this means we have to modify the implementation by directly changing and adapting the simulation code to our approach. An example, we will discuss here is the stochastic Galerkin method [135].

For non-intrusive methods, we do not need access to the implementation. However, we require knowledge about the underlying model, which we intent to exploit, e.g., the structure or properties of the PDEs, and adapt the method to the given problem. Reduced order methods are an example for problems in this field [65].

Lastly, for black-box methods, we just need to be able to evaluate the model in form of an input-to-output relationship. We do not require access to the equations and, in this work, also assume that we lack access to gradient information. In this context, we discuss for example stochastic collocation methods [330].

All approaches come, of course, with advantages and disadvantages. On the one hand, being able to modify the code in intrusive methods gives us the ability to create a highly specific and optimized solution for the problem, which usually results in high accuracy or computational efficiency. This, however, comes with the additional burden of modifying a potentially highly-complex software up to the point where this might be infeasible. Black-box methods, on the other hand, offer a simple solution to connect any application to our UQ method of choice, which makes implementation (usually) quite straight-forward. As a downside, the accuracy might be lower and computational cost are higher compared to the intrusive approach. Grey-box methods offer a compromise between the two extremes.

Intrusive approaches were made popular in the book by Ghanem and Spanos [135], which presented the *stochastic Galerkin* method. In this approach, the forward solution is projected onto a lower dimensional space. We pick the one-dimensional case for a simplified presentation.

A random variable $Q(x, \theta)$ can be expressed as an infinite sum

$$Q(x, \theta) = \sum_{i=0}^{\infty} \hat{q}_i(x) \phi_i(\theta). \tag{6.11}$$

Here, $\{\phi_i\}_{i=0}^{\infty}$ are orthonormal basis functions with respect to their corresponding probability density function $p(\theta)$, i.e., $< \phi_i(\theta), \phi_j(\theta) >= \int \phi_i(\theta)\phi_j(\theta)p(\theta)d\theta = \delta_{i,j}$. Eq. (6.11) is called *spectral projection* or *polynomial chaos expansion* (PCE) and $\{\hat{q}_i(x)\}_{i=0}^{\infty}$ are the *PCE coefficients*.

This expansion is approximated by using only N instead of an infinite number of coefficients and plug the PCE into the model equations that we want to solve. By using a spectral projection and the orthogonality of the system, we get a system of equations for N unknown coefficients $\{\hat{q}_i(x)\}_{i=0}^{N-1}$ that we need to solve. This requires, that we need to modify the simulation code accordingly. Given the coefficients, we can then solve for the statistics that we are interested in, e.g., the expected value or the variance:

$$\mathbb{E}[Q(x,\theta)] \approx \hat{q}_0(x) \qquad (6.12) \qquad \mathbb{V}[Q(x,\theta)] \approx \sum_{i=1}^{N-1} \hat{q}_i^2(x) \qquad (6.13)$$

In 1938, Wiener stated that random variables can be represented by orthonormal basis functions in his work on homogenous chaos [320]. This was proven by Cameron and Martin in [58], who show optimal convergence of the Hermite orthonormal basis in terms of Gaussian random variables. The authors Xiu and Karniadakis generalized the work in [331], where they presented a scheme to generate optimal orthonormal basis functions and the PCE from the Askey scheme. Ways to improve the computational performance of the resulting matrices have been investigated in [114].

This scheme was further generalized to arbitrary orthogonal polynomial basis (see, e.g. [328]). Introductions to the method are also given by Smith [290] and Sullivan [301]. Multiple different applications were explored since then, e.g., to stochastic diffustion [332], random ODEs [24] or elliptic PDEs [26, 219], the Boltzmann [163] and Navier-Stokes equations [294].

Another advantage of the stochastic Galerkin method is its close relation to the finite element method, where a lot of theory and analysis could readily be applied, e.g. analytic convergence results, see [37, 108, 332]. In the area of intrusive approaches, also other methods are investigated. More recently than stochastic Galerkin, and in the same spirit of projection methods, reduced basis approaches were explored in [65, 66], with applications in, e.g., elliptic PDEs [312] or optimal control [60].

As stated, black-box and grey-box approaches are non-intrusive and consider the underlying model (or simulation code) as (mostly) unknown. For both, the computation of statistics happens outside of the simulation code, as we have visualized in Fig. 1.2, and no modifications are required. We differentiate mainly between two classes of non-intrusive approaches—deterministic and sampling-based. This distinction concerns the choice of samples. In deterministic approaches, the choice of samples is predefined, e.g., by the underlying choice of interpolation or quadrature rule to solve the integral for the statistics. In sampling approaches, the choice of samples is (pseudo-)random and based on a (pseudo-)random number generator.

Possibly the most popular method in the area of deterministic non-intrusive methods is the *pseudo-spectral approach* [329], also based on spectral projection. Compared to stochastic Galerkin, we do not inject the pseudo-spectral projection in the model but project directly on the QoI to solve for the coefficients $\{\hat{q}_i(x)\}_{i=0}^{\infty}$. As a result, we can compute the coefficients as

$$\hat{q}_i(x) = \int_{\Omega} Q(x,\theta)\phi_i(\theta)p(\theta)d\theta. \tag{6.14}$$

This requires the approximation of an integral numerically using quadrature. The quadrature nodes and weights are chosen with respect to the distribution $p(\theta)$ such that

$$\hat{q}_i(x) \approx \sum_{j=0}^{N-1} Q(x, \theta_j)\phi_i(\theta_j)w_j, \tag{6.15}$$

where $\{\theta_j\}_{i=0}^{N-1}$ and $\{w_j\}_{i=0}^{N-1}$ are the quadrature nodes and weights, respectively.

With interpolation, we can also solve the integration problem using a cheaper surrogate. This approaches is called *stochastic collocation* [27, 330]. Here, we build an interpolation surrogate $\tilde{Q} \approx Q$ of the model, where $\tilde{Q}(x, \theta_i) = Q(x, \theta_i)$ at interpolation points $\{\theta_j\}_{i=0}^{N-1}$, which we can then evaluate and integrate at much lower computational cost. A common choice for the interpolation rule are, e.g. Lagrange polynomials.

The disadvantage of all the above approaches is their deterministic quadrature or interpolation rule where the samples stem typically from full-tensor grids. As a consequence, the number of quadrature or interpolation nodes grow exponentially with the number of dimensions. This is called the *curse of dimensionality* and results in high computational cost for large numbers of dimensions. Remedies exist to slow down the growth with the number of dimensions, e.g. sparse grids [56], as first presented in [242] with adaptive extensions in, e.g. [213, 214]. Nevertheless, with increasing dimensions, the curse of dimensionality is inevitable for these methods.

A cure for the curse of dimensionality are sampling methods whose number of samples do not depend on the dimension. In the next section, we present Monte Carlo sampling, the classic method in the field.

6.3 Monte Carlo sampling

Monte Carlo (MC) sampling [57, 226] is by far the most popular sampling algorithm for solving problems in forward UQ. Using a (pseudo-)random number generator to generate samples, we can approximate the expected value (or mean) of a QoI as

$$\widehat{\mu}_1[Q(\mathbf{x}, \boldsymbol{\theta})] := \frac{1}{N}\sum_{i=1}^{N} Q(\mathbf{x}, \boldsymbol{\theta}_i) = \frac{1}{N}\sum_{i=1}^{N} Q^{(i)}. \tag{6.16}$$

We assume that the samples $\{\boldsymbol{\theta}_i\}_{i=1}^{N}$ are *independent and identically distributed.*

We state and proof two well-known properties of this estimator in Eq. (6.16) in form of lemmas due to their importance in this work. We repeat: we assume that $Q(\mathbf{x}, \boldsymbol{\theta})$ itself is unbiased, meaning the high-fidelity solution is accurate enough for its discretization bias to be negligible.

The first property is with regards to the bias of the estimator.

Lemma 2. *The estimator* $\widehat{\mu}_1[Q(\mathbf{x}, \boldsymbol{\theta})]$*, as defined in Eq .*(6.16)*, is an unbiased estimator for* $\mathbb{E}[Q(\mathbf{x}, \boldsymbol{\theta})]$*:*

$$\mathbb{E}[\widehat{\mu}_1[Q(\mathbf{x}, \boldsymbol{\theta})]] = \mathbb{E}[Q(\mathbf{x}, \boldsymbol{\theta})]. \tag{6.17}$$

Proof.

$$\mathbb{E}[\widehat{\mu}_1[Q(\mathbf{x},\boldsymbol{\theta})]] = \mathbb{E}[\frac{1}{N}\sum_{i=1}^{N} Q^{(i)}] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[Q^{(i)}] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[Q] = \mathbb{E}[Q]. \tag{6.18}$$

□

The second property relates to the RMSE of the estimator. As we stated in Eq. (6.9), we know that for unbiased estimators, the RMSE corresponds to the variance of the estimator. In that spirit, we can compute the RMSE of Eq. 6.16.

Lemma 3. *The RMSE of the estimator $\widehat{\mu}_1[Q(\mathbf{x},\boldsymbol{\theta})]$ is given as*

$$\mathit{RMSE}[\widehat{\mu}_1[Q(\mathbf{x},\boldsymbol{\theta})]] = \sqrt{\frac{\mathbb{V}\left[Q\right]}{N}}. \tag{6.19}$$

Proof.

$$\mathrm{RMSE}[\widehat{\mu}_1[Q(\mathbf{x},\boldsymbol{\theta})]] = \sqrt{\mathbb{V}[\widehat{\mu}_1[Q(\mathbf{x},\boldsymbol{\theta})]]} = \sqrt{\mathbb{V}[\frac{1}{N}\sum_{i=1}^{N} Q^{(i)}]} = \sqrt{\frac{\mathbb{V}\left[Q\right]}{N}}. \tag{6.20}$$

□

This property of the RMSE for MC sampling shows both its advantage as well as disadvantage: as an advantage, the number of sampling points N do not directly depend on the dimension of the problem, d_{sto}. This is in stark contrast to the previously discussed deterministic non-intrusive methods that have to fight the curse of dimensionality where the growth of sampling points is often in $\mathcal{O}(N^{d_{\mathrm{sto}}})$ (or $\mathcal{O}(N\log(N)^{d_{\mathrm{sto}}-1}$ for sparse grid techniques). Additionally, the MC method is highly parallelizable, known as being *embarrassingly parallel.* The N evaluations of $\{Q^{(i)}\}_{i=1}^{N}$ are all independent from each other and can, in principle, be processed in parallel. Therefore, in principle, this method has a perfect strong scaling, depending on the available computational resources. However, the RMSE of the estimator decays slowly, proportionally to $N^{-1/2}$. Hence, to decrease the error of our estimator, the sample size has to be increased drastically. This slow convergence is still a main disadvantage of MC methods.

One direction of research to counteract the slow convergence are *quasi-Monte Carlo* (QMC) methods [57, 99]. Here, the main idea is not to use (pseudo-)random numbers to generate the random samples but instead to use a deterministic sequence instead. This sequence is picked in a way that it is *space filling*, meaning the distribution of the sample points in the space should be as uniform as possible. Using this approach, we can reach a convergence rate of $\mathcal{O}(\frac{\log(N)^{d_{\mathrm{sto}}}}{N})$. However, we again introduce a dependence on the dimension. Also, the performance of QMC heavily depends on properties of Q, e.g. its smoothness, which makes it often impracticable. The dependence on d_{sto} and its reduced simplicity and applicability reduce the charm of the method. New developments again introduce randomness in the squence construction (see [246] for an overview) but assumptions on the smoothness of the integrand still apply.

The other direction to improve MC is to use *variance reduction* methods [184]. Popular methods in this field are *importance sampling*, which has its main application in rare-event simulation [45] and *stratified sampling*, where the samples are picked according to space-filling properties [287], similar to QMC. However, both methods again suffer from their specificity to certain applications areas in UQ.

In the next section, we look at *multilevel* MC, an extension of standard MC sampling, if a discretization hierarchy is available. It has shown a wide range of applications and is one of the most popular techniques for sampling methods. It is also the main method in this work.

6.4 Multilevel methods

Multilevel MC (MLMC) was first introduced by [155] and reestablished by [136, 137] as an estimator for the expected value. This method needs to have access to a hierarchy of models that describe the same problem at different levels of accuracy and computational cost as a prerequiste. In multilevel methods, this is commonly realized by using different discretization levels of the model. Given such hierarchy, the idea is to combine the levels in an optimal way, employing sampling on multiple approximations or levels for a QoI to estimate the SoI introduced in Section 6.3. In the end, the goal is to only use a small amount of resources on the finest and most costly discretization while most evaluations are spent on lower levels. In the following, we introduce the approach for the mean estimator as described in [136, 155] following the notation we presented in our work in [223] and [224].

The methods assumes to have such a hierarchy of levels$\{Q_\ell\}_{\ell=1}^{L}$ available with associated, ordered computational cost $C_1 < C_2 < \cdots < C_L$ for a single evaluation. Here, we extend notation from the previous section to $Q_\ell := Q_\ell(\mathbf{x}, \boldsymbol{\theta}_\ell)$, where $\ell = 1$ denotes the coarsest and $\ell = L$ presents the finest (or high-fidelity) resolution. A realization (or sample) is then written as $Q_\ell^{(i)} := Q_\ell\left(\mathbf{x}, \boldsymbol{\theta}_\ell^{(i)}\right)$, where N_ℓ samples are used for level ℓ, as follows: $\left[Q_\ell^{(1)}, \dots, Q_\ell^{(N_\ell)}\right] := \left[Q_\ell(\mathbf{x}, \boldsymbol{\theta}_\ell^{(1)}), \dots, Q_\ell(\mathbf{x}, \boldsymbol{\theta}_\ell^{(N_\ell)})\right]$. Samples for Q_ℓ are obtained on different levels $\ell = 1, ..., L$, and, in particular, each multilevel estimator at level ℓ will include evaluations on two consecutive levels ℓ and $\ell - 1$.

The MLMC estimator for the mean of a QoI $Q := Q_L$ is expressed by expanding the expected value over levels, as

$$
\begin{aligned}
\mathbb{E}\left[Q_L\right] \approx \widehat{\mu}_{1,\mathrm{ML}}[Q_L] &:= \sum_{\ell=1}^{L} \widehat{\mu}_{1,\ell} = \sum_{\ell=1}^{L} \widehat{\mu}_1[Q_\ell - Q_{\ell-1}] \\
&= \sum_{\ell=1}^{L} \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} \left(Q_\ell^{(i)} - Q_{\ell-1}^{(i)}\right), \quad Q_0^{(i)} := 0,
\end{aligned}
\tag{6.21}
$$

On the one hand, the samples $\boldsymbol{\theta}_\ell^{(i)}$ between different levels ℓ of the sum are independent, hence there is no *inter-level* dependence for $\widehat{\mu}_{1,\ell}$. On the other hand, using the same samples $\boldsymbol{\theta}_\ell^{(i)}$ to compute the difference of terms in $\widehat{\mu}_1[Q_\ell - Q_{\ell-1}]$, introduces a correlation between the two quantities $Q_\ell^{(i)} := Q_\ell[\mathbf{x}, \boldsymbol{\theta}_\ell^{(i)}]$ and $Q_{\ell-1}^{(i)} := Q_{\ell-1}[\mathbf{x}, \boldsymbol{\theta}_\ell^{(i)}]$, resulting in an *intra-level* dependence. When we consider single-level estimators in the multilevel case, such as $\widehat{\mu}_{1,\ell}$, this inter-level independence and intra-level dependence is implicitly assumed for the rest of this work.

Let us look at properties of the MLMC estimator for the mean, where we start with its bias:

Lemma 4. *The MLMC estimator for the mean, as given in Eq.* (6.21)*, is an unbiased estimator,*

$$\mathbb{E}[\mu_{1,ML}[Q_L]] = \mathbb{E}\left[Q_L\right]. \tag{6.22}$$

Proof.

$$\begin{aligned}\mathbb{E}[\mu_{1,\mathrm{ML}}[Q_L]] &= \sum_{\ell=1}^{L} \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} \mathbb{E}\left[Q_\ell^{(i)} - Q_{\ell-1}^{(i)}\right] \\ &= \sum_{\ell=1}^{L} \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} (\mu_{1,\ell} - \mu_{1,\ell-1}) = \sum_{\ell=1}^{L} (\mu_{1,\ell} - \mu_{1,\ell-1}) \\ &= \mu_{1,L} = \mathbb{E}\left[Q_L\right].\end{aligned} \tag{6.23}$$

□

Next, we regard the RMSE. Due to the estimator in Eq. (6.21) being unbiased, its RMSE is a result of the estimator variance as we established in Lemma 1:

Lemma 5. *The RMSE of the MLMC estimator for the mean, as given in Eq.* (6.21)*, is*

$$RMSE[\widehat{\mu}_{1,ML}] = \sqrt{\mathbb{V}[\widehat{\mu}_{1,ML}]} = \sqrt{\sum_{\ell=1}^{L} \frac{1}{N_\ell} \mathbb{V}[Q_\ell - Q_{\ell-1}]}. \tag{6.24}$$

Proof.

$$\begin{aligned}\mathbb{V}[\widehat{\mu}_{1,\mathrm{ML}}] &= \sum_{\ell=1}^{L} \mathbb{V}\left[\widehat{\mu}_{1,\ell} - \widehat{\mu}_{1,\ell-1}\right] = \sum_{\ell=1}^{L} \mathbb{V}[\frac{1}{N_\ell} \sum_{i=1}^{N_\ell} (Q_\ell^{(i)} - Q_{\ell-1}^{(i)})] \\ &= \sum_{\ell=1}^{L} \frac{1}{N_\ell^2} \sum_{i=1}^{N_\ell} \mathbb{V}\left[Q_\ell^{(i)} - Q_{\ell-1}^{(i)}\right] = \sum_{\ell=1}^{L} \frac{1}{N_\ell} \mathbb{V}[Q_\ell - Q_{\ell-1}].\end{aligned} \tag{6.25}$$

□

To achieve the desired accuracy, the computational load can be redistributed toward the coarser level, Here, we assume that the variance over a sequence of levels decreases, such that $\mathbb{V}[Q_\ell - Q_{\ell-1}] \to 0$ with $\ell \to L$. In words: the variance of the difference between levels decreases for finer levels. To accomplish this redistribution, we also need to define an associated computational cost for each level, such that a single Q_ℓ evaluation has a computational cost of C_ℓ, and $C_1 < C_2 < \cdots < C_L$. This computational cost can, e.g., be related to the number of degrees of freedom or the computational runtime.

However, how to pick the number of samples $\mathbf{N} = [N_0, ..., N_\ell, ..., N_L]$ optimally for each level ℓ? The goal is to find an estimator with the smallest computational cost with a target estimator accuracy. The corresponding optimization problem can be written as

$$\begin{aligned} \overset{*}{\mathbf{N}}{}^{\mathbb{E}} &= \arg\min_{\mathbf{N}^{\mathbb{E}}} C_T^{\mathbb{E}} \\ \text{s.t. } \mathbb{V}[\widehat{\mu}_{1,\text{ML}}] &= \epsilon_{\mathbb{E}}^2, \end{aligned} \tag{6.26}$$

for a given target accuracy $\epsilon_{\mathbb{E}}^2$ where $C_T^{\mathbb{E}} := \sum_{\ell=1}^L C_\ell N_\ell^{\mathbb{E}}$ describes the total computational cost. We call this problem a *resource allocation* problem *targeting the mean.* When a distinction is required, we denote the target statistic either as superscript in the resource allocation, here $\overset{*}{\mathbf{N}}{}^{\mathbb{E}}$, or as subscript in the variance target, here $\epsilon_{\mathbb{E}}^2$. Doing so, we can differentiate between resource allocations optimized for specific statistics.

The MLMC estimator for the mean represents a special case in which the closed-form for its variance allows for a closed-form solution for the optimal sample distribution $\overset{*}{\mathbf{N}}{}^{\mathbb{E}}$. The resource allocation problem from Eq. (6.26) can be solved analytically as established in [136, 155].

Lemma 6. *The optimal resource allocation as solution of Eq.* (6.26) *for the MLMC estimator for the mean as presented in Eq.* (6.21) *is given as*

$$\overset{*}{N}{}_\ell^{\mathbb{E}} = \left\lceil \lambda \sqrt{\frac{\mathbb{V}[Q_\ell - Q_{\ell-1}]}{C_\ell}} \right\rceil, \tag{6.27}$$

where

$$\lambda = \epsilon_{\mathbb{E}}^{-2} \sum_{\ell=1}^L \sqrt{\mathbb{V}[Q_\ell - Q_{\ell-1}] C_\ell} \tag{6.28}$$

is the Lagrangian multiplier.

Proof. See A.1 for the proof. □

Remark 1. *We can also switch the optimization statement for the resource allocation. Instead of minimizing the cost for a target estimator variance, we can also reduce the estimator variance targeting a certain cost. This is useful if we do not have an estimate of the target accuracy available. Then, the resource allocation problem reads:*

$$\begin{aligned} &\min_{\mathbf{N}^{\mathbb{E}}} \mathbb{V}[\widehat{\mu}_{1,ML}], \\ s.t. \ &\sum_{\ell=1}^L C_\ell N_\ell^{\mathbb{E}} = C_T. \end{aligned} \tag{6.29}$$

where the method of Lagrange multipliers is given as

$$\lambda = C_T \sum_{\ell=1}^{L} \sqrt{\mathbb{V}[Q_\ell - Q_{\ell-1}] C_\ell}, \tag{6.30}$$

and finally the optimal resource allocation for each level is computed as

$$\overset{*}{N}{}_\ell^{\mathbb{E}} = \left\lceil \frac{1}{\lambda} \sqrt{\frac{\mathbb{V}[Q_\ell - Q_{\ell-1}]}{C_\ell}} \right\rceil . \tag{6.31}$$

MLMC has arguably become the most popular sampling method in UQ over the last decade. Multiple extensions to the method have been proposed since its ressurection in [136]. Especially when applying forward UQ methods for computational expensive simulations, the method can offer large computational savings (with the caveat that a hierachy has to be available).

We see first applications to elliptic PDEs in the work of [31] and [72]. In [73] and [96], the authors present extensions to continuous levels, i.e., if arbitrary or adaptive discretizations are possible. The MLMC method has seen many fields of application such as the computation of failure probabilities in [111], in the context of Bayesian optimization [140] or Bayesian inference [117]. Other advances and developments are presented in [169].

In our contribution, we look at MLMC estimators for higher-order moments. First developments in this area were done by Bierig et al. [40] who investigated unbiased estimators for the variance. In [189], Krumscheid et al. developed higher-order estimators using h-statistics and an approximation of the resource allocation problem, while in [188], they presented estimators for general functions. In the work by Ganesh et al. [127] and Ayoul-Guilmard et al. [25], the respective authors looked at MLMC estimators for robustness measures and the conditional-value-at-risk, specifically. We present our new contributions in this field, namely new and unbiased estimators for variance, the standard deviation and the linear combination of mean and standard deviation in Part III.

For completeness, but without going into mathematical details, we also mention *multi-index* and *multi-fidelity* methods, which can be seen as generalizations of the multilevel idea. Multi-index methods describe the idea of combining levels of discretizations in different dimensions [151]. For PDEs, we can, e.g., have discretizations in both space and time, and the question arises how to combine these discretizations optimally. Extensions of this work, look then in how to adaptively find the different levels [168]. Also adaptive sparse grids methods can be interpreted as a variant of an adaptive multiindex method [118]. In multi-fidelity methods, we combine different models to not only leverage different accuracy, but also different modelling choices with respect to the underlying physics. Approaches on how to create, select and arrange the model are given by the authors in [119, 142, 251]. We find applications of the approach in forward UQ [236, 241], OUU [240], variance reduction methods like importance sampling [250] but also in machine learning [161, 233] and Bayesian inference [143]. A comprehensive overview over multi-fidelity methods is given by Peherstorfer et al. [252] and recently by Fernández-Godino [121]. Finally, combinations of the multilevel and multi-fidelity method exist [110, 133], where we use multilevel estimators on the level hierarchy for each model, but also use a multi-fidelity method for a hierarchy of models. Just by the amount of articles in recent years, we can see that this is a field of high interest and many developments at the moment.

7 Gaussian processes

We introduce the background of *Gaussian processes* (GPs) here since they serve as a main component of SNOWPAC, the stochastic optimization method that we will introduce in Part IV and our second major contribution of this thesis. In the following, we will only give an overview of its properties, since a great introduction and detailed presentation is given in [269]. We will start off by introducing their background theory in the upcoming Section 7.1. Next, we introduce GP regression in Section 7.2, properties and examples of kernel functions in Section 7.3 and how to optimize hyperparameters in Section 7.4. We summarize results on GP posterior consistency, i.e. how the GP approximation improves with a growing amount of data in Section 7.5 and summarize this chapter in Section 7.6.

7.1 The statistical model and its notation

GPs have been experiencing a rise in popularity in the last decade in the machine learning community. As a powerful tool for flexible function approximation, they are used to solve supervised learning problems in regression and classification. However, GPs are not only a powerful tool in machine learning, but can also serve as general surrogate models to approximate arbitrary functions.

Generally, a GP is a statistical model that characterizes a distribution over functions within a continuous domain. Within this continuous space, each data point is represented as a normally distributed random variable defined by a mean and variance. As a result, any finite set of these random variables follows a multivariate normal distribution. The distribution of a GP represents the joint Gaussian distribution of all these random variables (which are infinitely many) and is determined by a covariance function (and its mean, which is commonly assumed to be zero). GPs can be regarded as a specific instance of general random fields (in space) or stochastic processes (in time), as they are confined to Gaussian distributions.

We introduce the notation for the mean m and covariance $\mathbb{C}\mathrm{ov}$ as the defining properties of the GP:

$$\begin{aligned} &m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \\ &\mathbb{C}\mathrm{ov}[f(\mathbf{x}), f(\mathbf{x}')] = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned} \tag{7.1}$$

where $\mathbf{x} \in X \subset \mathbb{R}^{d_{\det}}$.

A covariance function describes the relationship between two random variables. In our context, the covariance of f is defined by the function k as follows:

$$\mathbb{C}\mathrm{ov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}'). \tag{7.2}$$

The function $k : \mathbb{R}^{d_{\text{det}}} \times \mathbb{R}^{d_{\text{det}}} \to \mathbb{R}$ is called a *kernel*. The choice of kernel depends on the specific characteristics of f.

The covariance function (or kernel) enables us to define a distribution over functions and generate sample functions from it. If the random function f follows a GP distribution, we write

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{7.3}$$

For a GP with a specified kernel, we provide a practical guide on computing these sample functions in App. B.2.

Before we introduce GP regression next, we introduce the kernel matrix $\mathbf{K}(\mathbf{A}, \mathbf{B}) \in \mathbb{R}^{N \times M}$, where $\mathbf{A} \in \mathbb{R}^{N \times d_{\text{det}}}$ and $\mathbf{B} \in \mathbb{R}^{M \times d_{\text{det}}}$. This matrix is obtained by evaluating the kernel function $k(\mathbf{a}_i, \mathbf{b}_j)$ on the datasets $\mathbf{A}$ and $\mathbf{B}$, with row vectors $\mathbf{a}_i = [a_0, ..., a_{d_{\text{det}}}]^T = (\mathbf{A})_{i:}$ for $i = [1, ..., N]$ and $\mathbf{b}_j = [b_0, ..., b_{d_{\text{det}}}]^T = (\mathbf{B})_{j:}$ for $j = [1, ..., M]$. We will also use the shorthand notation $\mathbf{K}(\mathbf{A}, \mathbf{B}) = \mathbf{K}_{\mathbf{AB}}$ to refer to this kernel matrix. Furthermore, we adopt the abbreviated notation for function evaluations on a matrix. For instance, $f(\mathbf{A}) = \mathbf{y}$ indicates that evaluating the function f on each row vector $\mathbf{a}_i$ of $\mathbf{A}$ yields the corresponding output y_i, where i ranges from 1 to N. The resulting vector is then denoted as $\mathbf{y} = [y_0, ..., y_N]^T \in \mathbb{R}^N$.

7.2 Gaussian process regression

Given a training set comprising N training points stored row-wise in matrix notation, $\{\mathbf{x}_i\}_{i=1}^N = \mathbf{X} \in \mathbb{R}^{N \times d_{\text{det}}}$, with corresponding training labels $[y_0, ..., y_N]^T = \mathbf{y} \in \mathbb{R}^N$, regression aims to find a function $f : \mathbb{R}^{d_{\text{det}}} \to \mathbb{R}$ such that $f(\mathbf{X}) \approx \mathbf{y}$. This is in contrast to interpolation which looks for a function exactly matching the training data, $f(\mathbf{X}) = \mathbf{y}$. We do not want a strict equality as in interpolation, since we additionally assume that the function values $\mathbf{y}$ are subject to noise. This is modelled by a linear, Gaussian distributed error term $\boldsymbol{\epsilon}_{\text{GP}} \sim \mathcal{N}(0, \sigma_n^2 \mathbf{I})$ such that $\mathbf{y} = f(\mathbf{X}) + \boldsymbol{\epsilon}\text{GP}$ where $\sigma_n \in \mathbb{R}^N$ is a hyperparameter.

Hence, the task is to find the best function f. One approach to address this task is to restrict the pool of candidate functions. On one extreme end, there is classic linear regression, which limits the possible functions to be linear. However, this approach is rather restrictive, lacking the freedom to explore other potential fits and potentially hinders discovering the best solution. The other extreme end involves testing all possible functions and selecting the best fit based on the training data. However, due to the infinite function space, this approach will be computationally expensive or even infeasible.

To tackle the challenge, GPs offer an appealing solution: they represent a distribution over functions, where the only constraints arise from the kernel function and its hyperparameters. By leveraging GPs, we can explore a wide range of functions. This allows for more flexibility in finding the best fit compared to linear regression, without reaching the extreme computational cost of testing all possible functions.

To narrow down the pool of candidate functions, we utilize the training points as constraints. The goal is to find the best function $f : \mathbb{R}^N \to \mathbb{R}$ to compute function values f^* for a set of test (or regression) points $\mathbf{X}^*$ conditioned on the noisy training points $(\mathbf{X}, \mathbf{y})$.

Now, to find the best function f conditioned on the noise training points $(\mathbf{X}, \mathbf{y})$, this leads us to the following model (refer to [47, 269] for more details):

We know that the function f we search is conditioned on the data points. This results in the *prior* distribution:

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}_{\mathbf{XX}}), \tag{7.4}$$

where $\mathcal{N}$ denotes the Gaussian distribution with zero mean and covariance $\mathbf{K}_{\mathbf{XX}}$. Without loss of generality, we assume a zero mean for the prior.

We introduce noisy evaluations as $\mathbf{y}$ by adding the hyperparameter σ_n^2 in the variance:

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_{\mathbf{n}}^{\mathbf{2}}\mathbf{I}). \tag{7.5}$$

Note, that we assume independent noise, which results in the off-diagonal entries to be zero.

Following the rules in App. B.3, we can compute the *likelihood*

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X}) p(\mathbf{f}|\mathbf{X}) d\mathbf{f} = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{\mathbf{XX}} + \sigma_{\mathbf{n}}^{\mathbf{2}}\mathbf{I}). \tag{7.6}$$

We assume noisy observations $(\mathbf{X}, \mathbf{y})$ as training data for the regression, but in the end are interested in a prediction f^* for an unseen point $\mathbf{x}^*$. Hence, we introduce the joint distribution between the noisy training evaluations and non-noisy test points:

$$\begin{bmatrix} \mathbf{y} \\ f^* \end{bmatrix} \sim \mathcal{N}(\mathbf{y}, f^*|\mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{XX}} + \sigma_{\mathbf{n}}^{\mathbf{2}}\mathbf{I} & \mathbf{k}_{\mathbf{X}\mathbf{x}^*} \\ \mathbf{k}_{\mathbf{x}^*\mathbf{X}} & k_{\mathbf{x}^*\mathbf{x}^*} \end{bmatrix}). \tag{7.7}$$

Following Eq. (B.17) and the rules of conditional Gaussian distributions in App. B.4 we condition the joint distribution to contain only those functions which agree with the observed data pairs $(\mathbf{X}, \mathbf{y})$. We present the *posterior* which gives us a prediction for f^* given a new data point $\mathbf{x}^*$:

$$\begin{aligned} f^*|\mathbf{x}^*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}(\mathbf{f}|&\mathbf{k}_{\mathbf{x}^*\mathbf{X}}[\mathbf{K}_{\mathbf{XX}} + \sigma_{\mathbf{n}}^{\mathbf{2}}\mathbf{I}]^{-1}\mathbf{y}, \\ &k_{\mathbf{x}^*\mathbf{x}^*} - \mathbf{k}_{\mathbf{x}^*\mathbf{X}}[\mathbf{K}_{\mathbf{XX}} + \sigma_{\mathbf{n}}^{\mathbf{2}}\mathbf{I}]^{-1}\mathbf{k}_{\mathbf{X}\mathbf{x}^*}). \end{aligned} \tag{7.8}$$

This results in a closed form posterior where we use the mean value as the estimator for a new data point $\mathbf{x}^*$:

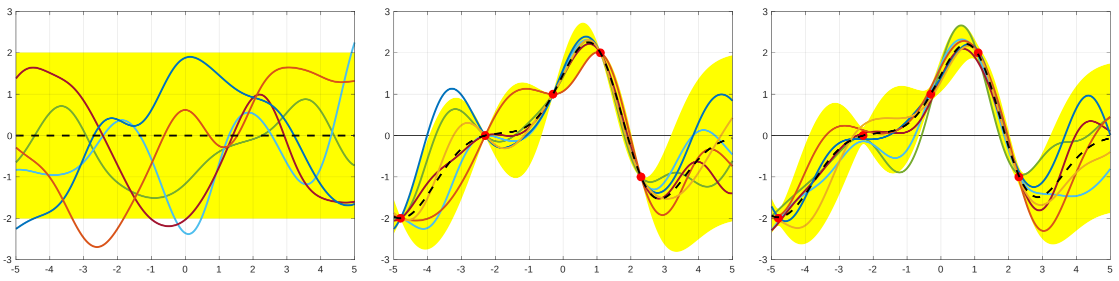
$$f^* = \mathcal{G}[\mathbf{x}^*|\mathbf{y}, \mathbf{X}] := \mathbf{k}_{\mathbf{x}^*\mathbf{X}}[\mathbf{K}_{\mathbf{XX}} + \sigma_{\mathbf{n}}^{\mathbf{2}}\mathbf{I}]^{-1}\mathbf{y}. \tag{7.9}$$

Additionally, the posterior offers an estimate about the uncertainty of its prediction through its variance:

$$\sigma_{f^*}^2 = \mathcal{V}[\mathbf{x}^*|\mathbf{X}] := k_{\mathbf{x}^*\mathbf{x}^*} - \mathbf{k}_{\mathbf{x}^*\mathbf{X}}[\mathbf{K}_{\mathbf{XX}} + \sigma_{\mathbf{n}}^{\mathbf{2}}\mathbf{I}]^{-1}\mathbf{k}_{\mathbf{X}\mathbf{x}^*}. \tag{7.10}$$

Here, we introduce the notation for the GP mean estimator and variance estimator, $\mathcal{G}$ and $\mathcal{V}$ respectively.

Example 1 (GP prior and GP posterior with and without noise)**.** *In Fig. 7.1, we show a comparison between drawing functions from a prior GP (left) and the corresponding posterior conditioned on given data points. On the one hand, if we do not apply noise the posterior functions interpolate the training points (center). On the other hand, if we assume noisy evaluations, they do not exactly match the training data (though they will converge to them with more training data) (right). We also observe how the mean GP estimator changes when being trained on the (noisy) data, (not) matching the data points. Additionally, we see a difference in the standard deviation estimator of the GP, with a standard deviation of 0 for the interpolated training data.*



(a) Prior. **(b)** Posterior. **(c)** With noise.

Figure 7.1: Five random Gaussian functions drawn from GP prior with squared exponential kernel (left). Six Gaussian functions sampled from posterior fitted on five training points (center). Six Gaussian functions sampled from posterior fitted on five noisy training points (right). We also plot the GP mean estimator, $\mathcal{G}$, in black dashed and two times the standard deviation, $\mathcal{V}^{\frac{1}{2}}$, as yellow band.

7.3 Kernel functions: definition, properties and examples

The choice of kernel function is fundamental in GP regression, since it builds the base of the function space that we want to represent. Kernel functions were initially introduced in [8] for potential function methods in pattern recognition. Subsequently, their popularity was revived in machine learning for optimal margin classifiers by [53]. Kernel functions capture our assumptions about the target function we aim to learn and the set of functions we consider. This is akin to classic regression, where we restrict the possible functions to linear or quadratic forms, for example. However, kernel functions can be much more powerful since we only describe the relation between pairs of data points of the function and do not restrict the function itself. In that sense, they provide information about the similarity between pairs of data points $(\mathbf{x}, \mathbf{x}')$.

In the context of supervised learning, this notion is intuitive since similar points should exhibit high correlation and be close to each other. Conversely, widely separated points should have minimal or no influence on each other. Consequently, training points that are in close proximity to a given test input provide the most information for estimation. However, not all functions can serve as kernel functions. In the following discussion, we present a few properties and examples of kernel functions, with a more comprehensive exploration available in [47, 159, 269].

As stated before, a kernel represents the relation of two data points $\mathbf{x} \in \mathcal{X}$ and $\mathbf{x}' \in \mathcal{X}$ from a data set $\mathcal{X} \subset \mathbb{R}^{d_{\text{det}}}$. Its formulation is given as

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}'). \tag{7.11}$$

Here, $\phi(x)$ describes a *nonlinear feature space* mapping. Interpreting Eq (7.11), the scalar product of function mappings ϕ applied on $\mathbf{x}$ and $\mathbf{x}'$ can be replaced by a kernel k applied on both vectors. This is called the *kernel trick* or *kernel substitution* [47].

We summarize a few more properties about kernels: If a kernel is invariant to translations in the input space, meaning it only depends on the difference $(\mathbf{x} - \mathbf{x}')$, it is referred to as *stationary.* A kernel is considered *symmetric* if it satisfies the property $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$. Additionally, if the kernel is solely a function of the distance $r = |\mathbf{x} - \mathbf{x}'| = \sqrt{(x_1 - x_1') + (x_2 - x_2') + ... + (x_{d_{\text{det}}} - x_{d_{\text{det}}}')}$, it is termed *isotropic.* Kernels of this type are also known as *radial basis functions* (RBF) or RBF kernels. A kernel is considered *positive semidefinite* if

$$\int k(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mu(\mathbf{x}) d\mu(\mathbf{x}') \geq 0, \tag{7.12}$$

for all $f \in L_2(\mathcal{X}, \mu)$. A kernel can be defined by its pairs of eigenvalues $\{\lambda_i\}_{i=1}^{\infty}$ and eigenfunctions $\{\phi_i\}_{i=1}^{\infty}$ through Mercer's theorem [185]:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i^*(\mathbf{x}'). \tag{7.13}$$

We call a kernel *(non-)degenerate* depending on it (not) containing eigenvalues equal to zero.

The matrix $\mathbf{K}$ is commonly referred to as the *Gram matrix* in literature [47, 269]. It is constructed from a given set of input points $\{\mathbf{x}_i\}_{i=1}^{N}$, where the entries are defined as $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. If the kernel is positive semidefinite, the resulting kernel matrix is, thus, also *positive semidefinite*, i.e. the inequality $\mathbf{v}^T \mathbf{K} \mathbf{v} \geq 0$ holds for all $\mathbf{v} \in \mathbb{R}^N$. It is important to note that the terms *Gram matrix*, *kernel matrix*, and *covariance matrix* are used interchangeably in this context. A more comprehensive discussion on the topic of properties of kernel and the kernel matrix can be found in references such as [47, 269].

The most prominent kernel used throughout literature is called *squared exponential kernel*, given by:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{\Lambda}_l^{-1} (\mathbf{x} - \mathbf{x}')\right). \tag{7.14}$$

Here, $\sigma_f^2 \in \mathbb{R}_+$ and $\mathbf{l} = \text{diag}(\mathbf{\Lambda}_l)$ with $\mathbf{l} = [l_1, ..., l_{d_{\text{det}}}]^T \in \mathbb{R}_+^{d_{\text{det}}}$ are called *hyperparameters*. Below, we present a selection of other kernels commonly found in the literature [47, 159, 269]. The table displays these kernels, their corresponding mathematical expressions and hyperparameters

Name	Expression	Hyperparameters
Constant	σ_0^2	σ_0
Linear	$\sum_{d=1}^{D} \sigma_d^2 x_d x_d'$	σ_d
Polynomial	$(\mathbf{x} \cdot \mathbf{x}' + \sigma_0^2)^p$	σ_0
Squared exponential	$\exp\left(-\frac{r^2}{2l^2}\right)$	l
Exponential	$\exp\left(-\frac{r}{l}\right)$	l
γ-exponential	$\exp\left(-(\frac{r}{l})^\gamma\right)$	l, γ
Rational quadratic	$(1 + \frac{r^2}{2\alpha l^2})^{-\alpha}$	α, l
Neural network	$\sin^{-1}(\frac{2\tilde{\mathbf{x}}^T \Sigma \tilde{\mathbf{x}}'}{\sqrt{(1+2\tilde{\mathbf{x}}^T \Sigma \tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}^T \Sigma \tilde{\mathbf{x}}')}})$	Σ
Wiener	$\min(\mathbf{x}, \mathbf{x}')$	
Brownian Bridge	$\min(\mathbf{x}, \mathbf{x}' - \mathbf{x}\mathbf{x}')$	
Matérn	$\frac{1}{2^{\nu-1}\Gamma(\nu)}(\frac{\sqrt{2\nu}}{l} r)^\nu \mathbf{K}_\nu(\frac{\sqrt{2\nu}}{l} r)$	ν, l
Ornstein-Uhlenbeck	$\exp\left(\frac{-r}{l}\right)$, D=1	l

Table 7.1: Examples of GP kernels, their names, expressions and hyperparameters.

7.4 Gaussian process hyperparameters

When discussing *hyperparameters* in the context of GP regression, we are referring to the parameters within the kernel function and the noise estimate in our linear noise model. For instance, in the case of the squared exponential kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{\Lambda}_l^{-1} (\mathbf{x} - \mathbf{x}')\right), \tag{7.15}$$

we have the variance parameter $\sigma_f^2 \in \mathbb{R}_+$ and the diagonal length-scale matrix $\mathbf{\Lambda}_l \in \mathbb{R}_+^{d_{\text{det}} \times d_{\text{det}}}$, which contains the length-scale vector $\mathbf{l} = [l_1, ..., l_{d_{\text{det}}}]^T$ along its diagonal. Thus, the complete hyperparameter space for the GP can be summarized as:

$$\boldsymbol{\psi} = [\sigma_f^2, l_1, ..., l_{d_{\text{det}}}] \in \mathbb{R}_+^{(d_{\text{det}}+1)}. \tag{7.16}$$

In some cases, authors also treat the noise estimate $\sigma_n^2 \in \mathbb{R}_+^N$ from the linear noise model $\mathbf{y} = f(\mathbf{x}) + \epsilon_{\text{GP}}, \epsilon_{\text{GP}} \sim \mathcal{N}(0, \sigma_n^2)$ as a hyperparameter [269, 293, 314]. We exclude this parameter in the hyperparameter optimization process, since, in our specific case, the MC estimate in SNOWPAC already provides a noise estimate for each training point. This will be explained in detail in Part IV. We give two examples on the effect of the length scale and the noise hyperparameters on the GP estimator next.

Example 2 (GP hyperparameters: length scale and noise)**.** *We present a comparison demonstrating the impact of the hyperparameters, namely the length scale l and σ_f^2, on the GP approximation for a one-dimensional regression problem in Fig. 7.2 and Fig. 7.3. For both figures, the training data consists of noisy samples of the test function* $\frac{\sin x}{x}$*, indicated by red points, which are uniformly sampled in the interval* $[-10, 10]$*. The test* *point range is $[-15, 15]$. The exact function f is represented by the blue dashed line. The GP mean value evaluated on the test set is depicted by the black line. Additionally, the shaded yellow region represents the 95% confidence interval for the standard deviation.*

In Fig. 7.2, we compare different length scales, $l \in \{0.3, 1.0, 5.0\}$. The results reveal a smoothing effect as the length scale increases. While the case of $l = 0.3$ attempts to fit closely to all the training points, for $l = 5.0$ we see a much smoother function but less fit to individual points.

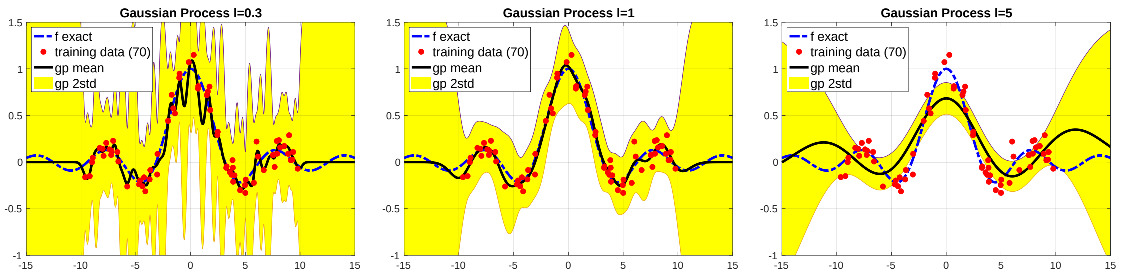


Figure 7.2: We compare the GP approximation using three different length scales $l \in \{0.3, 1.0, 5.0\}$ for the squared exponential kernel from left to right.

In Fig. 7.3, we illustrate the influence of σ_f^2 on the width of the prior variance. We compare three different values for $\sigma_f^2 \in \{0.1, 1, 5\}$. Smaller values of σ_f^2 introduce additional smoothing effects to the estimate, while a larger value allows larger variations.

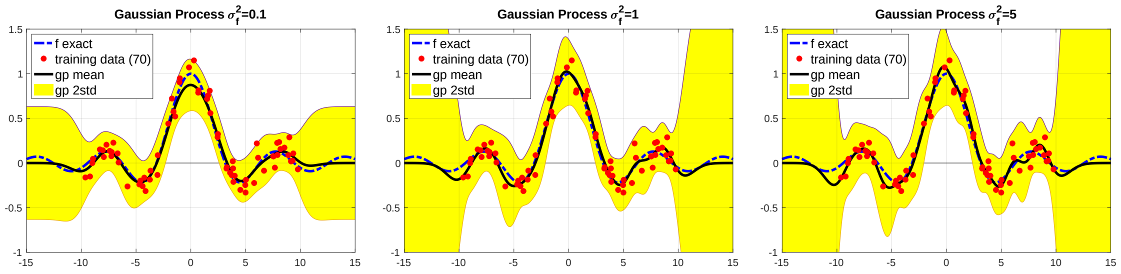


Figure 7.3: We compare the GP approximation using three different values for $\sigma_f^2 \in \{0.1, 1, 5\}$ for the squared exponential kernel from left to right.

The last question of this section is: How do we pick the optimal set of hyperparameters? While other approaches, such as cross validation [28, 218] and approaches for sparse data [216], exist, the most common approach is to minimize the negative marginal log-likelihood as given in (7.6). Here, we optimize for $\boldsymbol{\psi} = [\sigma_n, \mathbf{l}]$ as our vector of design parameters. We can write the marginal likelihood in its Gaussian form as

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}) &= \mathcal{N}(y|0, \mathbf{K_{XX}} + \sigma_n^2 I) \\ &= \frac{1}{\sqrt{2\pi\frac{d_{\det}}{N}|\mathbf{K_{XX}} + \sigma_n^2\mathbf{I}|}} \cdot \exp\left(-\frac{1}{2}\mathbf{y}^T(\mathbf{K_{XX}} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}\right). \end{aligned} \tag{7.17}$$

The logarithm is applied to simplify the formulation, removing the exponential:

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{X}) &= \log 1 - \log \sqrt{2\pi \frac{d_{\text{det}}}{N} |\mathbf{K_{XX}} + \sigma_n^2 \mathbf{I}|} - \frac{1}{2}\mathbf{y}^T (\mathbf{K_{XX}} + \sigma_{\mathbf{n}}{}^2 \mathbf{I})^{-1}\mathbf{y} \\ &= -\frac{1}{2}\log 2\pi \frac{d_{\text{det}}}{N} - \frac{1}{2}\log |\mathbf{K_{XX}} + \sigma_n^2 \mathbf{I}| - \frac{1}{2}\mathbf{y}^T (\mathbf{K_{XX}} + \sigma_n^2 \mathbf{I})^{-1}\mathbf{y}.\end{aligned} \tag{7.18}$$

Finally, since we are only interested in the maximal value, we can ignore constant terms in the optimization process and end up with the following optimization problem:

$$\min_{\boldsymbol{\psi}} - \log p(\mathbf{y}|\mathbf{X}) = \frac{1}{2}\log |\mathbf{K_{XX}} + \sigma_n^2 \mathbf{I}| + \frac{1}{2}\mathbf{y}^T (\mathbf{K_{XX}} + \sigma_n^2 \mathbf{I})^{-1}\mathbf{y}. \tag{7.19}$$

Here, $|\mathbf{A}|$ denotes the determinant of matrix $\mathbf{A}$.

7.5 Posterior consistency of Gaussian processes

Posterior consistency is a property of a sequence of probability distributions derived from a sequence of increasing datasets. More formally, if we say that a statistical procedure is *posterior consistent*, we mean that as the sample size tends to infinity, the posterior distribution of the parameters concentrates more and more around the true value of the parameter.

For GPs, the concept of posterior consistency becomes important when we want to make inferences about our data. If a GP is posterior consistent, then as we gather more and more data, the posterior distribution of our predictions will converge to the true underlying function that we are trying to estimate. Posterior consistency for GPs is tied to assumptions about the kernel function used, the noise in the data, and the true underlying function that generated the data. If these assumptions are satisfied, then a GP model can be shown to be posterior consistent.

Work in [280, 299, 319] present results on GP posterior consistency. However, these results assume deterministic data, i.e. $\sigma_n^2 = 0$, where they apply analysis from radial basis functions. More recent work by Lederer et al. [198] presents probabilistic uniform error bounds. We cite here Theorem 3.3 from their work and adapt it to our notation where necessary:

Consider a zero mean GP defined through the continuous covariance kernel $k(\cdot,\cdot)$ with Lipschitz constant L_k on the set $\in X \subset \mathbb{R}^{d_{det}}$. Furthermore, consider an infinite data steam of observations $(\mathbf{x}_i, y)$ of an unknown function $f : X \to \mathbb{R}$ with Lipschitz constant L_f and maximum absolute value $\bar{f} \in \mathbb{R}_+$ on X, where unknown function f is a sample from a GP $\mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$ and observations $y = f(\mathbf{x}) + \epsilon$ are perturbed by zero mean i.i.d. Gaussian noise ϵ with variance σ_n^2. Let $\mathcal{G}$ and $\mathcal{V}^{\frac{1}{2}}$ denote the mean and standard deviation of the GP conditioned on the first N observations. If there exists a $\epsilon > 0$ such that the standard deviation satisfies $\mathcal{V}^{\frac{1}{2}} \in \mathcal{O}\bigg(\log(N)^{-\frac{1}{2}-\epsilon}\bigg)$, $\forall\, \mathbf{x} \in X$, then it holds for every $\delta \in (0,1)$ that

$$P\bigg(\sup_{\mathbf{x}^* \in \mathbb{X}} ||\mathcal{G}[\mathbf{x}^*|\mathbf{y}, \mathbf{X}] - f(\mathbf{x}^*)|| \in \mathcal{O}(\log(N)^{-\epsilon})\bigg) \geq 1 - \delta. \tag{7.20}$$

For more details that go beyond what is relevant here, we refer to their paper, where the authors, e.g., discuss how to estimate the Lipschitz constants L_k and L_f or how to find the constant $\bar{f}$. The assumptions are not very restrictive, with a continuous kernel and a bounded function f. For us, importantly, this theorem shows that the probabilistic error of the GP mean estimator converges with an increasing number of observations N in the limit depending on the noise ϵ. We give an intuition about this property numerically in the following small example.

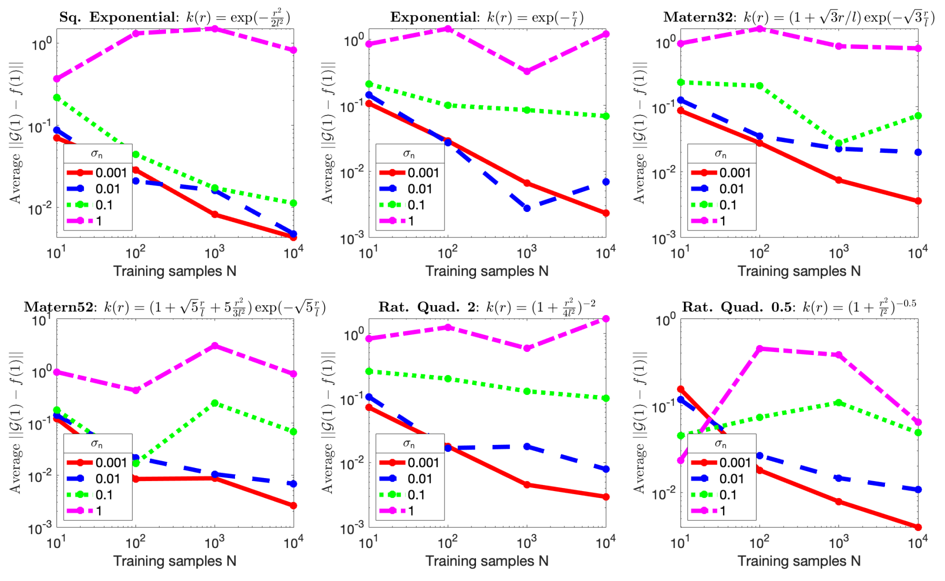


Figure 7.4: We compare the GP convergence by calculating the GP approximation error with respect to the function value at $x = 1$ for an increasing number of training samples. The error is averaged over 100 runs. We juxtapose different kernels given in the title and different noise levels, $\sigma_n \in \{1, 0.1, 0.01, 0.001\}$, in each figure.

Example 3 (GP posterior consistency example)**.** *We plot the numerical convergence of the GP for the test function $f(x) = \frac{\sin x}{x}$ in Fig. 7.4. We evaluate $f(x)$ for $x = 1$ for an increasing size of training data, $N \in \{10, 100, 1000, 10000\}$. The training data is picked randomly and uniformly around $x = 1$. For each training size, we evaluate the approximation error $||\mathcal{G}[\mathbf{x}^*|\mathbf{y}, \mathbf{X}] - f(\mathbf{x}^*)||$ and repeat this process 100 times, plotting the average approximation error. We compare different kernels, presented in Table 7.1, where the respective kernel is given in the title of the figures. For each kernel, we regard four different noises $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ on the data for $\sigma_n \in \{1, 0.1, 0.01, 0.001\}$. For each kernel and each training size, we optimize for the hyperparameters l and σ_f^2, minimizing the negative marginal log-likelihood.*

We observe a decrease in the approximation error in almost all cases. On the one hand, we notice, that convergence improves for smaller noises. On the other hand, the convergence rate seems to decrease for larger noise values, especially for the case

$\sigma_n = 1$. *Thus, a small noise value seems vital for the convergence of the GP, confirming the results from Thm. 7.20.*

Another interesting quantity to look at for convergence is the variance estimator of the GP, $\mathcal{V}^{\frac{1}{2}}(x)$ (see Eq. (7.10)). This estimator can be interpreted as the confidence of the GP as an estimator given the data. This estimator is independent of the function values, only depending on the data locations. Hence, we expect it to decrease it with an increasing amount of training samples. We numerically test this in the following example, where we look at the same setting as in Example 3, only changing the quantity of interest to $\mathcal{V}^{\frac{1}{2}}(x)$.

Example 4 (GP variance convergence example)**.** *We regard at the GP standard deviation $\mathcal{V}^{\frac{1}{2}}(x)$ for $x = 1$ for the same function $f(x) = \frac{\sin x}{x}$. We plot the GP estimate in Fig. 7.5. We evaluate again for an increasing size of training data, $N \in \{10, 100, 1000, 10000\}$ picked randomly and uniformly around $x = 1$ and repeat this process 100 times. We compare the same kernels and noise levels, and optimize for the hyperparameters l and σ_f^2, minimizing the negative marginal loglikelihood.*

For all cases, we observe (almost) linear convergence in the GP standard deviation estimate for increasing number of training samples. This seems independent of the noise level with some outliers for $\sigma_n = 1$. Note again, that this result is independent of the function f. Thus, the GP gets more confident in its estimation for larger N.

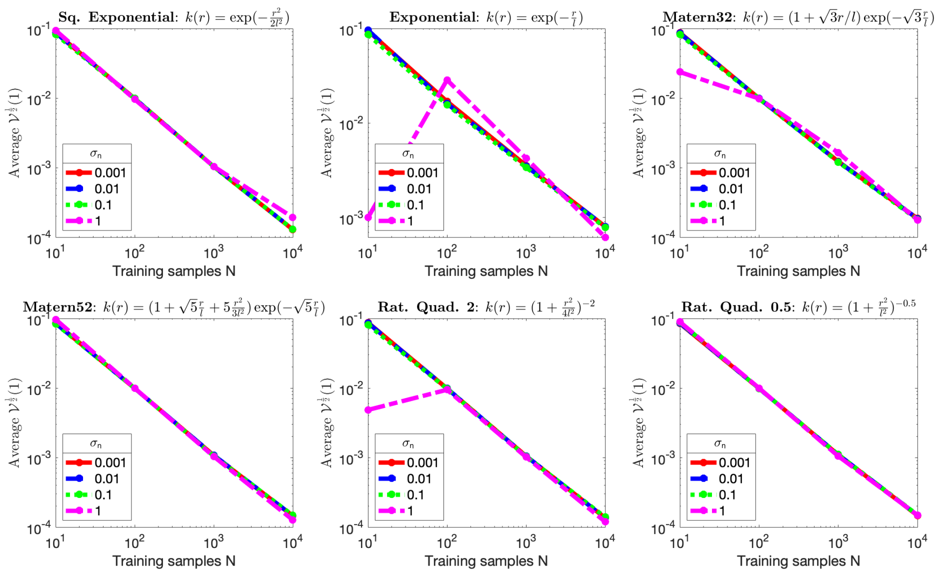


Figure 7.5: We compare the GP standard deviation estimate, $\mathcal{V}^{\frac{1}{2}}(x)$, for an increasing number of training samples. The error is averaged over 100 runs. We juxtapose different kernels given in the title and different noise levels, $\sigma_n \in \{1, 0.1, 0.01, 0.001\}$, in each figure.

Summarizing these results, we observe that the GP mean and standard deviation estimator improve with an increasing amount of training data. We showed both theoretical and numerical results here. This property will be important when we talk about GP surrogates in Part IV, where we employ the method in our optimization method SNOWPAC.

7.6 Preview of Gaussian processes in our work

GPs are crucial to this work. They are a main component in SNOWPAC, which we will present in Part IV, where they serve as a surrogate to mediate the effect of noisy evaluations. The main function of GP surrogates in SNOWPAC is presented in Section 14.3. Here, the posterior consistency of the GP is an important property. Furthermore, the choice of kernel constrains the functions the GP can represent. In additional extensions to the method, we will also look into adaptive kernel choices in SNOWPAC in Section 17.1. Given that the posterior evaluation of the GP involves an inversion of the kernel matrix, which can become computationally expensive, we will present approximate GP methods as an extension in SNOWPAC in Section 17.2.

8 Derivative-free deterministic optimization

The next two topics of this background part concern optimization. In this chapter, we talk about the deterministic case before we extend to OUU in the following chapter. We will start off with an in-depth review of developments in the field given the importance of the topic for our thesis in Section 8.1. We will present the basics of derivative-free optimization and different approaches to solve such problems in Section 8.2. We will close this chapter in Section 8.3, where we present the deterministic derivative-free optimization method NOWPAC, which our contribution extends on.

8.1 A brief review of optimization

The website by Mitri Kitti [183] provides a comprehensive overview of breakthroughs in the field of optimization spanning over the past 2300 years, revealing that optimization problems have been explored since ancient times.

One of the earliest optimization problems documented in print is found in Euclid's renowned work, Elements [245]. Euclid formulated the problem of determining the maximum area of a parallelogram, given a triangle, such that one of the parallelogram's edges is parallel to an edge of the triangle. Euclid also introduced other optimization problems, including the search for the minimum distance between a point and a line, as well as the demonstration that a rectangle with maximum area is a square [303]. Zenodorus, another Greek mathematician, studied Dido's problem, which involved finding the object with the largest area bounded by a given perimeter [167].

In terms of significant advancements in the field, there is a noticeable gap in the subsequent centuries until the 17th and 18th centuries, when new works emerged. Johannes Kepler, for example, investigated the possibility of packing more balls per unit volume than the conventional packing method [95]. This problem arose when Sir Walter Raleigh sought advice on the most efficient way to stack cannonballs on his ship. Although the initial answer may appear straightforward[1], it was not until Carl Friedrich Gauss that significant progress was made towards proving the optimal solution [130]. In fact, the final proof was only published in this century by Thomas C. Hales [152].

Other renowned scientists, who delved into optimization include Galileo Galilei, who studied the shape of a hanging chain [149], Isaac Newton, who investigated bodies of minimal resistance [288], Joseph-Louis Lagrange, who examined minimal surfaces [100], and Adrien-Marie Legendre, who presented the least-squares method [225].

[1]Make layers in which each cannonball touches six others, and stack these layers such that the cannonball in one fits into the holes between those below.

Optimization appears in virtually every field and application. Its applications span finance [85], economics [88], biology and medicine [153], engineering [249], and even sports [283].

8.2 Deterministic derivative-free optimization

Let us first define the general deterministic optimization problem to set the stage:

$$\begin{aligned} &\min f(\mathbf{x}) \\ \text{s.t.} \quad &c_i(\mathbf{x}) \leq 0, i = 1, ..., r. \end{aligned} \tag{8.1}$$

We denote $\mathbf{x} \in \mathbb{R}^{d_{\text{det}}}$ as the *design parameters*, $f(\mathbf{x}) : \mathbb{R}^{d_{\text{det}}} \to \mathbb{R}$ as the scalar-valued *objective function* and $\{c_i(\mathbf{x}) : \mathbb{R}^{d_{\text{det}}} \to \mathbb{R}\}_{i=1}^{r}$ as the set of *constraint functions.* We define the *feasible domain* as $X := \{\mathbf{x} \in \mathbb{R}^{d_{\text{det}}} : c(\mathbf{x}) \leq 0\}$. Note that we define Eq. (8.1) as a minimization problem though it can be easily changed to a maximization problem, since $\min f(\mathbf{x}) = -\max f(\mathbf{x})$.

Extensive literature exists on various optimization approaches depending on the characteristics of the objective and constraints functions. Examples include linear optimization [92, 93], convex optimization [55], and nonlinear optimization [36]. In our approach, we do not impose specific restrictions on the problem and consider it to be nonlinear.

Handling constraints is a critical aspect of nonlinear optimization problems, and numerous approaches have been explored in literature. One approach involves replacing the objective function with a merit function that penalizes constraint violations [36, 55, 78]. To ensure the effectiveness of these methods, appropriate penalization parameters need to be carefully selected. Alternatively, a filter technique proposed by [122] and [123] allows for iterative optimization steps to be accepted, if they satisfy either the objective function or the constraints. However, there are other methods that do not rely on filter techniques or penalty approaches as, e.g. presented in [145]. When function evaluations are computationally expensive, reduced-order models can be employed to reduce computational effort, as discussed in [6]. Despite their differences, most of these methods still rely on derivative information regarding the objective function and constraints.

A category of optimization solvers that can handle Eq. (8.1) without relying on gradient information are referred to as *derivative-free* methods [81, 192]. These solvers become necessary when there either is no knowledge of the analytical objective function or it is not available in closed form. Therefore, derivative-free methods have a broader applicability but also present their own set of challenges. The most prominent (and obvious) challenge is the absence of gradient access, which requires finding an alternative approach to determine the optimal direction.

We will now summarize two popular classes of methods to solve derivative-free optimization problems [81]: direct-search methods and line-search methods. Afterwards, we will talk in more extend about trust-region methods for derivative-free optimization, which we employ in our optimization method SNOWPAC.

Direct-search methods

Direct-search (or pattern-search) methods use evaluations of the objective function at a finite number of points in each iteration to decide which action to take solely based on those function values [202, 203, 261]. The first work on this method is attributed to Fermi and Metropolis [120] back in 1952, with a recent review given in [13].

There is no explicit derivative approximation or model building involved in contrast to what we see in the other classes of methods. They usually consist of a search step and a poll step. The search step is optional and evaluates the objective function at a finite number of points (usually arranged in a mesh or pattern, but it can also follow some heuristic). If the search step does not find an improved iterate, i.e. $f(\mathbf{x}_{k+1}) \geq f(\mathbf{x}_k)$, the poll step is performed: it consists of a local search around the current iterate, exploring a set of points defined by a step size parameter and a positive basis. The step size parameter is adapted based on the success of the poll step. The simplest example for a positive basis is the coordinate-search method [172], which samples possible new iterations using the unit vectors, where all entries of the vector are zero apart from the i-th direction. The point, which improves the objective the most, becomes the new iterate.

Due to their simplicity, direct-search methods offer great analytical properties [306] and are often provably globally convergent [16]. The special class of mesh adaptive direct search (MADS) methods were shown to achieve global convergence also in the case of a nonsmooth objective [17]. A popular method in this field is NOMAD (Nonlinear optimization with the MADS algorith) [194], which is still being developed [21] and has in the meantime been extended to constrained cases [19, 75]. MADS methods have also been applied to mixed variable optimization [2] and multi-objective optimization [22]. Their main advantages are their simplicity and traceability, but also their potential for computational optimization, e.g., evaluating the poll or search step in parallel.

Another very popular algorithm is the Nelder-Mead algorithm [237]. Instead of a grid based approach, Nelder-Mead-type algorithms use a simplex based on $d_{\text{det}} + 1$ vertices to search the area. The simplex is then transformed—reflected, expanded or contracted—based on the success of the search. It can also be shown to be globally convergent [307] and is very popular still to date, despite being first introduced in the sixties due to its simplicity.

Line-search methods

In line-search methods, we use approximate gradient information [81, 139]. Given a sample set of $d_{\text{det}} + 1$ points, we can compute the derivative using linear interpolation (assuming that the points are distributed well). If we have $(d_{\text{det}} + 1)(d_{\text{det}} + 2)/2 - 1$ points available, we can even approximate second order information by computing the Hessian. Given the gradient direction, methods in this class perform a line-search to find the next best point along that direction. This class of methods is also shown to be globally convergent [52]. To give a few examples of developments in the field: the authors [212, 148] present connections with direct-search methods; we see modifications of the basic algorithm to parallel computations [128] as well as a quite recent extensions to multi-fidelity methods [253].

Trust-region methods

A widely used category of methods in derivative-free optimization is known as *trust-region methods*, which developments were presented in a variety of reviews over the years [78, 81, 176, 192, 231, 334]. In trust-region methods, a surrogate model is constructed in the vicinity of the current iterate, satisfying specific properties. The region of vicinity is called the *trust-region* and is usually a d_{det}-dimensional sphere (or sometimes a box [239]). This surrogate model is then used to approximate derivative information. The next iterate is found by solving a (simpler) optimization problem on the subproblem defined by the surrogate in the trust-region where said derivative information is available. The iterate is then accepted based on the *acceptance ratio*, which is a criteria of quality of the surrogate. Closeness to a first order local optimum is based on a *criticality measure*, which shows convergence of the solver.

Formally, let $\mathbf{x}_0$ denote the initial iterate, followed by a series of intermediate steps $\mathbf{x}_k, k \in \mathbb{N}_0$. Given an objective function f and constraints $c_i, i = 1, ..., r$, we construct surrogates $m^f_{\mathbf{x}_k}$ and $m^{c_i}_{\mathbf{x}_k}$ in the neighborhood of the current iterate $\mathbf{x}_k$. This neighborhood called trust-region is defined as $B(\mathbf{x}_k, \rho_k) := \mathbf{x} \in \mathbb{R}^D : \|\mathbf{x} - \mathbf{x}_k\| \leq \rho_k$, where $\rho_k \in \mathbb{R}_+$ represents the adaptive trust-region radius for each iteration $k \in \mathbb{N}_0$.

The surrogate itself is based on quadratic models, which makes it particularly mathematically attractive due to its curvature information. Two typically required properties of the surrogate are that it is twice continuous differentiable and *fully linear*, which is defined as (see [81]):

$$\begin{aligned} |f(\mathbf{x}+\mathbf{s}) - m^f_{\mathbf{x}}(\mathbf{x}+\mathbf{s})| &\leq \kappa_f \rho^2 \\ |c_i(\mathbf{x}+\mathbf{s}) - m^{c_i}_{\mathbf{x}}(\mathbf{x}+\mathbf{s})| &\leq \kappa_c \rho^2 \\ \|\bigtriangledown f(\mathbf{x}+\mathbf{s}) - \bigtriangledown m^f_{\mathbf{x}}(\mathbf{x}+\mathbf{s})\| &\leq \kappa_{df} \rho \\ \|\bigtriangledown c_i(\mathbf{x}+\mathbf{s}) - \bigtriangledown m^{c_i}_{\mathbf{x}}(\mathbf{x}+\mathbf{s})\| &\leq \kappa_{dc} \rho \end{aligned} \tag{8.2}$$

.

Another important aspect regarding the quality of the surrogate is how *well-poised* its surrogate points are. In other words, the quality of the surrogate model in the trust-region is determined by the position of the underlying points. For example, if a model $m^f_{\mathbf{x}_k}$ tries to extrapolate a function value f at points far away from its interpolation points, the model value may differ greatly from the value of f. Λ-*poisedness* is a concept to measure how well a set of points is dispersed through a region of interest, and ultimately how well a model will estimate the function in that region.

Remark 2. ***Λ-poisedness:*** *The most commonly utilized measure for assessing the spatial distribution of points within a specific region of interest is based on Lagrange polynomials. Let $\mathbf{X} = \{\mathbf{x_1}, ..., \mathbf{x_p}\}$ represent a set of p points. A set of Lagrange* *polynomials forms a basis that satisfies the following conditions:*

$$l_j(\mathbf{x}_i) = \begin{cases} 1 & \textit{if } i = j, \\ 0 & \textit{if } i \neq j. \end{cases} \tag{8.3}$$

A set of points **X** *is considered* Λ*-poised on a set* B *if* **X** *is linearly independent, and the Lagrange polynomials* $l_1, ..., l_p$ *associated with* **X** *satisfy the inequality [82]:*

$$\Lambda \geq \max_{1 \leq i \leq p} \max_{\mathbf{x} \in B} |l_i(\mathbf{x})|. \tag{8.4}$$

with $\Lambda_{min} := \max_{1 \leq i \leq p} \max_{\mathbf{x} \in B} |l_i(\mathbf{x})|$. *It is important to note that this definition is independent of the function being modeled, and the points* **X** *are not necessarily required to be elements of the set* B. *Additionally, if a model is poised on a set* B*, it is also poised on any subset of* B. *Our main interest lies in determining the smallest possible value of* Λ *that satisfies Eq.* (8.4).

We can then construct a quadratic surrogate as

$$m_k^b(\mathbf{x}_k + \mathbf{s}) = m_k^b(\mathbf{x}_k) + \mathbf{g}_k \mathbf{s} + \frac{1}{2}\mathbf{s}^T \mathbf{H}_k \mathbf{s} + q_k \tag{8.5}$$

for a general function $b \in \{f, c_1, ..., c_r\}$, with the interpolation condition $m_i^b(\mathbf{x}_i) = b(\mathbf{x}_i), i = 0, ..., k$. We denote $\mathbf{g}_k \in \mathbb{R}^{d_{\text{det}}}$ and $\mathbf{H}_k \in \mathbb{R}^{d_{\text{det}} \times d_{\text{det}}}$ as the gradient vector and the symmetric Hessian matrix, respectively, and $q_k \in \mathbb{R}$ as a constant term. The vector $\mathbf{s} \in \mathbb{R}^{d_{\text{det}}}$ is the step size restricted by the trust-region $||\mathbf{s}_k||_2 \leq \rho_k$, where $|| \cdot ||_2$ is the Euclidean norm. We need $(d_{\text{det}}+1)(d_{\text{det}}+2)/2$ points to fully define the model. If the model is underdefined, we can use *minimum Frobenius norm* surrogates to minimize the approximation error based on the Frobenius norm of the Hessian $\mathbf{H}_k$ [258]. Here, we find the Hessian by requiring again the interpolation condition, $m_i^b(\mathbf{x}_i) = b(\mathbf{x}_i), i = 0, ..., k$, and optimize for the surrogate that minimizes $||\mathbf{H}_k - \mathbf{H}_{k-1}||_F^2$. This approach requires at least $(d_{\text{det}} + 1)$ points, to start with a linear polynomial.

From the surrogate, we can then derive the gradients, since

$$\begin{aligned} \nabla m_k^b(\mathbf{x}_k + \mathbf{s}) &= \mathbf{g}_k + \mathbf{H}_k \mathbf{s}, \\ \nabla^2 m_k^b(\mathbf{x}_k + \mathbf{s}) &= \mathbf{H}_k, \end{aligned} \tag{8.6}$$

and optimize for the next iterate using a subproblem optimizer of our choice.

Since the surrogate is simple and well-understood with clear analytic properties, trust-region methods offer great analytic properties as well, e.g. with respect to provability of convergence [76, 79, 80]. Polynomial response surfaces [80, 256, 257, 258] and radial basis functions [322] are common examples for typically employed surrogate models. For a more detailed discussion of surrogate model selection, we point to [81].

Compared to line-search methods, trust-region methods do not search at first for a direction and a step length but rather restrict the search space. The idea of minimizing a quadratic interpolation within some region of validity goes back to the authors of [326, 327]. With respect to important developments in this field, the authors of [257] introduced a method using quadratic surrogates in an unconstrained scenario and later-on introduced least-Frobenius norm to adaptively update the surrogate [258]. In [61], the authors investigated the effect of inexact gradient values, which is of relevance when dealing with noisy function evaluations. The authors of [173] looked at quadratic models for noisy function evaluations. Random trust-region approaches, investigated in [67] and [191], build on the idea the condition of using fully linear model as surrogates need to hold only with some probability, as was shown in [30] with extension in [147]. Also, radial basis functions were investigated as alternative surrogate models in [321, 322] with an extension to constrained problems in [272].

Otherwise, regarding trust-region methods for constrained problem, the literature is not as rich. Apart from the method NOWPAC [23], which we will discuss next, the authors of [259] present a fast method for linear constraints, whereas the authors in [74] defer the handling of constraints to the subproblem optimizer inside the trust region. In [14], Audet et al. present a trust-region algorithm using a progressive barrier, which is an approach to push the infeasible solution (progressively) toward the feasible domain.

8.3 Review of the trust-region framework NOWPAC

In the previous section, we discussed various derivative-free algorithms, with a particular emphasis on trust-region methods. Now, we will provide a brief overview of NOWPAC (Nonlinear Optimization With Path-Augmented Constraints), which is deterministic derivative-free optimization method, presented by Augustin and Marzouk [23]. NOWPAC utilizes a trust-region approach. It employs black-box evaluations to construct fully-linear surrogate models of the objective function.

NOWPAC offers three contributions to the field of deterministic derivative-free optimization using trust regions:

1. It employs an inner boundary path to ensure feasibility by convexifing the constraint locally.
2. The algorithm is provably convergent to a first-order critical point.
3. NOWPAC provides an error indicator to detect corrupted evaluations, e.g. due to noise.

The algorithm starts at an initial feasible solution from $\mathbf{x}_0 \in X$, where the feasible domain is given by $X = \left\{\mathbf{x} \in \mathbb{R}^{d_{\text{det}}} : c_i(\mathbf{x}) \leq 0 \text{ for } i = 1, ..., r, \right\}$ with an initial trust-region size ρ_0. Next, it builds the initial surrogates m_0^b by evaluating d_{det} points around $\mathbf{x}_0$. Here, it uses the unit vectors as stencil in each dimension for initial points $\mathbf{x}_i = \mathbf{x}_0 + 0.5\rho_0 e_i$ for $i = 1, ..., d_{\text{det}}$.

At optimization iteration $k \in \mathbb{N}^+$, NOWPAC computes the criticality measure as a first step:

$$\alpha_k(\rho_k) := \frac{1}{\rho_k} \left| \min_{\substack{\mathbf{x}_k + \mathbf{d} \in X_k \\ \|\mathbf{d}\| \leq \rho_k}} \left\langle \mathbf{g}_k^f, \mathbf{d} \right\rangle \right| . \tag{8.7}$$

It is used to assess closeness of the trial step to a first-order optimal point—a *critical point*. Here, $\mathbf{g}_k^f = \nabla m_k^f$ is the gradient of the surrogate model of the objective function f as given in Eq. 8.6. If the criticality measure is lower than some preset tolerance, NOWPAC could stop, saying that it has reached an optimum; however, the algorithm assumes that the surrogate does not have the best predictive capabilities. Instead of stopping, it reduces the trust region and updates the surrogates. The algorithm is therefore stopped, if the trust-region radius falls below some specified tolerance.

If this tolerance has not been reached yet, an intermediate step $\mathbf{s}_k$ to the next best optimum is computed by solving the trust-region subproblem

$$\begin{aligned} \mathbf{s}_k &:= \arg\min m_k^f(\mathbf{x} + \mathbf{s}_k) \\ \text{s.t.} \quad & \mathbf{x} \in X_k, \ \|\mathbf{s}_k\| \le \rho_k \end{aligned} \tag{8.8}$$

with the approximated feasible domain

$$X_k := \left\{ \mathbf{x} \in \mathbb{R}^n \ : \ m_k^{c_i}(\mathbf{x}) + h_k(\mathbf{s}_k) \le 0 \text{ for } i = 1, ..., r, \right\}. \tag{8.9}$$

The additive offset h_k to the constraints is called the inner boundary path, a convex offset-function to the constraints ensuring convergence of NOWPAC. It can be interpreted as a penalty or barrier function to push the design away from the constraint.

From the step, NOWPAC computes the trial point $\tilde{\mathbf{x}}_{k+1} = \mathbf{x}_k + \mathbf{s_k}$. As a first criteria, it only accepts this trial step if it is feasible with respect to the exact constraints $\{c_i\}_{i=1}^r$, i.e. if $\{c_i \le 0\}_{i=1}^r$. Otherwise, the trust-region radius is reduced and, after having ensured fully linearity of the models m_k^f and $\{m_k^{c_i}\}_{i=1}^r$, the algorithm starts from the beginning and a new trial step $\mathbf{s_k}$ is computed.

If the point is feasible, the acceptance ratio is computed as

$$r_k = \frac{f(\mathbf{x}_k) - f(\tilde{\mathbf{x}}_{k+1})}{m_k^f(\mathbf{x}_k) - m_k^f(\tilde{\mathbf{x}}_{k+1})}, \tag{8.10}$$

to assess acceptance of the trial point and to update of the trust region

This ratio reflects the truth versus the prediction of the surrogate. Hence, it is a measure for the quality of our surrogate. Based on the result, for some tolerance η_1, NOWPAC accepts or rejects the point

$$\mathbf{x}_{k+1} = \begin{cases} \tilde{\mathbf{x}}_{k+1} & \text{if} \quad r_k \ge \eta_1 \\ \mathbf{x}_k & \text{if} \quad r_k < \eta_1 \end{cases}. \tag{8.11}$$

In the final step of the iteration, it adapts the trust-region accordingly

$$\rho_{k+1} = \begin{cases} \rho_k & \text{if} \quad r_k \ge 2 \\ \gamma_{inc}\rho_k & \text{if} \quad r_k \ge \eta_2 \quad (and) \quad r_k < 2 \\ \rho_k & \text{if} \quad \eta_1 \le r_k < \eta_2, \\ \gamma_{dec}\rho_k & \text{if} \quad r_k < \eta_1. \end{cases} \tag{8.12}$$

Note that when NOWPAC accepts the suggested next iterate $\tilde{\mathbf{x}}_{k+1}$, it either keeps the trust-region radius or increases it, while it decreases the radius if it rejects the suggested iterate. This represents a balance between exploration and exploitation. If NOWPAC accepts a point, it can explore further to find a better minimum and trusts its surrogate. If it rejects the point, it has to exploit the current region and has to improve the surrogate by decreasing its volume. We summarize the simplified algorithm for NOWPAC in Algorithm 1 [23].

Algorithm 1 Simplified NOWPAC

```
1: Construct the initial fully linear models $m_0^f(\mathbf{x}_0 + \mathbf{s})$, $\{m_0^{c_i}(\mathbf{x}_0 + \mathbf{s})\}_{i=1}^r$, k = 0
2: while $\rho_k >= \rho_{min}$ do
3:    Compute criticality measure $\alpha_k(\rho_k)$ via (8.7)
4: // STEP 0: Criticality step
5:    while $\alpha_k(\rho_k)$ is too small do
6:       Decrease $\rho_k = \omega\rho_k$ and update $m_k^f$ and $\{m_k^{c_i}\}_{i=1}^r$
7:    end while
8: // STEP 1: Step calculation
9:    Compute a trial step $\mathbf{s}_k$ via (8.9)
10: // STEP 2: Check feasibility of trial point
11:    if $\exists c_i(\mathbf{x}_k)(\mathbf{x}_k + \mathbf{s}_k) > 0$ for $i = 1, ..., r$ then
12:       Set $\rho_k = \gamma\rho_k$ and update $m_k^f$ and $\{m_k^{c_i}\}_{i=1}^r$
13:       Go to STEP 0
14:    end if
15: // STEP 3: Acceptance of trial point and update trust-region
16:    Compute $r_k$ via (8.10)
17:    if $r_k \geq \eta_0$ then
18:       Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$
19:       Include $\mathbf{x}_{k+1}$ into the node set and update the models to $m_{k+1}^f$ and $\{m_{k+1}^{c_i}\}_{i=1}^r$
20:    else
21:       Set $\mathbf{x}_{k+1} = \mathbf{x}_k$, $m_{k+1}^f = m_k^f$ and $\{m_{k+1}^{c_i} = m_k^{c_i}\}_{i=1}^r$
22:    end if
23:    Update $\rho_{k+1}$ via (8.12)
24:    Update $m_{k+1}^f$ and $\{m_{k+1}^{c_i}\}_{i=1}^r$
25:    k = k+1
26: end while
```

By presenting the background about derivative-free optimization and the trust-region method NOWPAC, we lay the groundwork of this thesis for optimization. A major question of our second contribution, SNOWPAC, is how to extend NOWPAC for stochastic parameters and thus developing a derivative-free optimization method for OUU. Next, we formulate the foundations of OUU in the next chapter and subsequently present our contribution of SNOWPAC in Part IV.

9 Optimization under uncertainty

OUU can be viewed as a generalization from deterministic optimization, where we add random variables to our optimization problem. Due to the introduced randomness, these problems are also referred to as *stochastic optimization.* Since we are looking for a robust solution with respect to these uncertainties, another common denotation is *robust optimization.*

In the field of OUU, randomness is introduced into the previously deterministic problem from Eq 8.1. The resulting OUU problem can be written as

$$\begin{aligned} &\min \mathcal{R}^f(\mathbf{x},\boldsymbol{\theta}),\\ \text{s.t.}\quad &\mathcal{R}^{c_i}(\mathbf{x},\boldsymbol{\theta}) \leq 0, i = 1,...,r. \end{aligned} \tag{9.1}$$

Here, $\mathcal{R}^f$ and $\{\mathcal{R}^{c_i}\}_{i=1}^r$ are *measures* of *robustness* or *risk.* Apart from the deterministic design variable $\mathbf{x}$, we re-introduce the random variable $\boldsymbol{\theta}$, as presented in Chap. 6. The challenge is now to compute the measures, which are usually some form of statistics such as the expected value. In the next Section 9.1, we are going to discuss common measures. We will conclude this chapter with a review of work in the field in Section 9.2. This chapter is extending our work in [221].

9.1 Problem formulations for optimization under uncertainty

In the subsequent discussions, we will delve into commonly used measures of robustness and risk and their corresponding sampling estimators. This is a crucial aspect of OUU, as these measures provide a quantitative assessment of the risk associated with a particular decision or design. We refer to [275, 302] for a detailed discussion about risk assessment strategies.

We introduce a collection of measures $\mathcal{R}^f$ and $\{\mathcal{R}^{c_i}\}_{i=1}^r$ to model robustness and risk for the OUU problem of Eq. (9.1). To simplify the notation, we refer to the objective function f and the constraints $\{c_i\}_{i=1}^r$ as black box b and the corresponding measures will be denoted by $\mathcal{R}^b$. We further assume that b is square integrable with respect to $\boldsymbol{\theta}$, i.e. its variance is finite, and its cumulative distribution function is continuous and invertible at every fixed design point $\mathbf{x} \in \mathbb{R}^{d_{\text{det}}}$. In other words, we assume to be able to sample $\boldsymbol{\theta}$.

The classical first example for a robustness measure is the expected value

$$\mathcal{R}_0^b(\mathbf{x}) := \mathbb{E}\left[b(\mathbf{x};\boldsymbol{\theta})\right] = \int_\Omega b(\mathbf{x};\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}. \tag{9.2}$$

While it may be debated whether the expected value accurately reflects robustness concerning variations in $\boldsymbol{\theta}$ (given that it does not provide insight into the spread of b around $\mathcal{R}_0^b(\mathbf{x})$), it remains a widely utilized measure for handling uncertainty in optimization problems. For instance, the expected objective value, $\mathcal{R}_0^f$, yields an average-optimized design, while $\{\mathcal{R}_0^{c_i}\}_{i=1}^r$ indicates expected feasibility.

To capture the statistical distribution of b around $\mathcal{R}_0^b$ for various values of $\boldsymbol{\theta}$—thereby providing a justification for the term 'robustness measure'—we introduce a standard deviation term:

$$\mathcal{R}_1^b(\mathbf{x}) := \mathbb{V}^{\frac{1}{2}}\left[b(\mathbf{x};\boldsymbol{\theta})\right] = \sqrt{\mathbb{E}\left[b(\mathbf{x};\boldsymbol{\theta})^2\right] - \mathcal{R}_0^b(\mathbf{x})^2} = \sigma\left[b(\mathbf{x};\boldsymbol{\theta})\right]. \tag{9.3}$$

An important measure is the linear combination of $\mathcal{R}_0^b$ and $\mathcal{R}_1^b$, which we denote as *scalarization*:

$$\mathcal{R}_2^b(\mathbf{x}) := \gamma q_1 \mathcal{R}_0^b(\mathbf{x}) + (1-\gamma) q_2 \mathcal{R}_1^b(\mathbf{x}), \tag{9.4}$$

which offers a natural decision-making interpretation. The measure $\mathcal{R}_2^b(\mathbf{x})$ balances the dual objectives of expected outcome minimization and reduction of the range of potential outcomes. By minimizing the standard deviation term $\mathcal{R}_1^b$, we enhance our confidence in the optimal value being aptly represented by $\mathcal{R}_0^b$. The trade-off between these two potentially conflicting goals is mediated by the weight factor $\gamma \in [0,1]$, reflecting the user's preference. The constants q_1 and q_2 are necessary to achieve a proper scale balance between $\mathcal{R}_0^b$ and $\mathcal{R}_1^b$.

A standard practice is to employ measures known as probabilistic constraints—or chance constraints—as defined in [263]. Here, a probability level $\beta \in ;]0,1[$ is determined, up to which the optimal design should remain feasible. The measure that corresponds to this is given as:

$$\mathcal{R}_3^{b,\beta}(\mathbf{x}) := \mathbb{E}\left[\mathbf{1}(b(\mathbf{x},\boldsymbol{\theta}) > 0)\right] - (1-\beta), \tag{9.5}$$

where $\mathbf{1}$ is the indicator function

$$\mathbf{1}(x) = \begin{cases} 1 \text{ if } x \text{ is True,} \\ 0 \text{ if } x \text{ is False.} \end{cases} \tag{9.6}$$

These probabilistic constraints are used in economic modeling, for instance, ensuring that the construction cost of a power plant do not exceed a certain budget with a probability of β. In physics, an application would be adjusting the gas mixture in a combustion chamber to avoid flame extinction with a (high) probability of β. Penalties related to costs or risks from violating these constraints can be included in the objective function. We refer to the work by Li et al. [205, 206, 207] for an efficient method to approximate $\mathcal{R}_3^{b,\beta}$.

Assuming the existence of an invertible cumulative distribution function, F_μ, probabilistic constraints can be articulated in terms of quantile functions:

$$\mathcal{R}_4^{b,\beta}(\mathbf{x}) := \min\left\{\alpha \in \mathbb{R} \,:\, \mathbb{E}[b(\mathbf{x},\boldsymbol{\theta}) \leq \alpha] \geq \beta\right\}. \tag{9.7}$$

Although the two formulations $\mathcal{R}_3^{b,\beta}$ and $\mathcal{R}_4^{b,\beta}$ yield the same set of feasible points, $\{\mathbf{x} \in \mathbb{R}^{d_{\text{det}}} \,:\, \mathcal{R}_3^{c,\beta}(\mathbf{x}) \leq 0\} = \{\mathbf{x} \in \mathbb{R}^{d_{\text{det}}} \,:\, \mathcal{R}_4^{c,\beta}(\mathbf{x}) \leq 0\}$, we demonstrate in App. D.1

that $\mathcal{R}_4^{b,\beta}$ often presents more favourable smoothness properties than $\mathcal{R}_3^{b,\beta}$, making it a more suitable model for probabilistic constraints in our optimization process. For $b = f$, the robustness measure $\mathcal{R}_4^{f,\beta}$ is also known as Value at Risk (VaR), a widely adopted non-coherent risk measure in finance applications. Also, note that if the underlying distribution is (assumed to be) normal, $\mathcal{R}_2^b(\mathbf{x})$ is often used in practice as a chance constraint, where $\gamma c_1 = 1$ and $(1-\gamma)c_2$ describes the confidence interval.

The Conditional Value at Risk (CVaR), as detailed in Acerbi [3] and Rockafeller [275], is a coherent extension of $\mathcal{R}_4^{b,\beta}(\mathbf{x})$. This measure is characterized as the conditional expectation of b exceeding the VaR:

$$\mathrm{CVaR}_\beta(\mathbf{x}) := \frac{1}{1-\beta} \int\limits_{b(\mathbf{x},\boldsymbol{\theta}) \geq \mathcal{R}_4^{b,\beta}(\mathbf{x})} b(\mathbf{x},\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{9.8}$$

Taking cue from Alexander et al. [9] and Rockafeller et al. [275], we define the last robustness measure of this work as

$$\mathcal{R}_5^{b,\beta}(\mathbf{x},\tau) := \tau + \frac{1}{1-\beta}\mathbb{E}\left[\max\{b(\mathbf{x},\boldsymbol{\theta}) - \tau, 0\}\right]. \tag{9.9}$$

This definition allows us to minimize the CVaR without the need to first compute $\mathcal{R}_4^{b,\beta}$, as the minimization of $\mathcal{R}_5^b$ over the extended feasible domain $X \times \mathbb{R}$ yields

$$\min_{\mathbf{x}\in X} \mathrm{CVaR}_\beta(\mathbf{x}) = \min_{(\mathbf{x},\tau)\in X\times\mathbb{R}} \mathcal{R}_5^b(\mathbf{x},\tau). \tag{9.10}$$

To be thorough in this work, we note another measure traditionally closely associated with robust optimization: the worst case formulation given as $\max_{\boldsymbol{\theta}}\{b(\mathbf{x},\boldsymbol{\theta})\}$. This measure, however, can often pose computational challenges. We basically search a rare event, where its analytical computation is only computationally feasible in particular situations with simpler, non-black-box functions. Importantly, it does not necessitate an understanding of the probability distribution of $\boldsymbol{\theta}$, a condition we assume to be true in this work, since we are just looking for the maximum value. As such, we will not discuss worst-case scenario approaches further in this context.

We give an example for the different measures $\mathcal{R}_3^{b,\beta}$, $\mathcal{R}_4^{b,\beta}$ and $\mathcal{R}_5^{b,\beta}$ next. This helps in giving an intuition of the optimization formulations in the stochastic setting. Afterwards, we move on to a literature review about work in the field of OUU.

Example 5 (Illustration for measures)**.** *We plot examples of $\mathcal{R}_0^{b,\beta}$, $\mathcal{R}_1^{b,\beta}$, $\mathcal{R}_3^{b,\beta}$, $\mathcal{R}_4^{b,\beta}$ and $\mathcal{R}_5^{b,\beta}$ for $\beta = 0.95$ for a one-dimensional Gaussian distribution $b(\cdot,\theta) \sim \mathcal{N}(-1,1)$ in Fig. 9.1. We point out the quantities such as α in $\mathcal{R}_4^{b,\beta}$ or CVaR in $\mathcal{R}_5^{b,\beta}$ in the figures. We also stress again the similarity of $\mathcal{R}_3^{b,\beta}$ and $\mathcal{R}_4^{b,\beta}$. While in $\mathcal{R}_3^{b,\beta}$ (left), we compare the two areas under the curve (red and blue), we optimize for α such that the red area equals β for $\mathcal{R}_4^{b,\beta}$ (center). Hence, we see a match of the blue vertical line in the left plot with the red vertical line in the central plot. Finally, the CVaR in the right plot denotes the expected value of the red area under the curve, i.e. of all values that are larger than the VaR.*

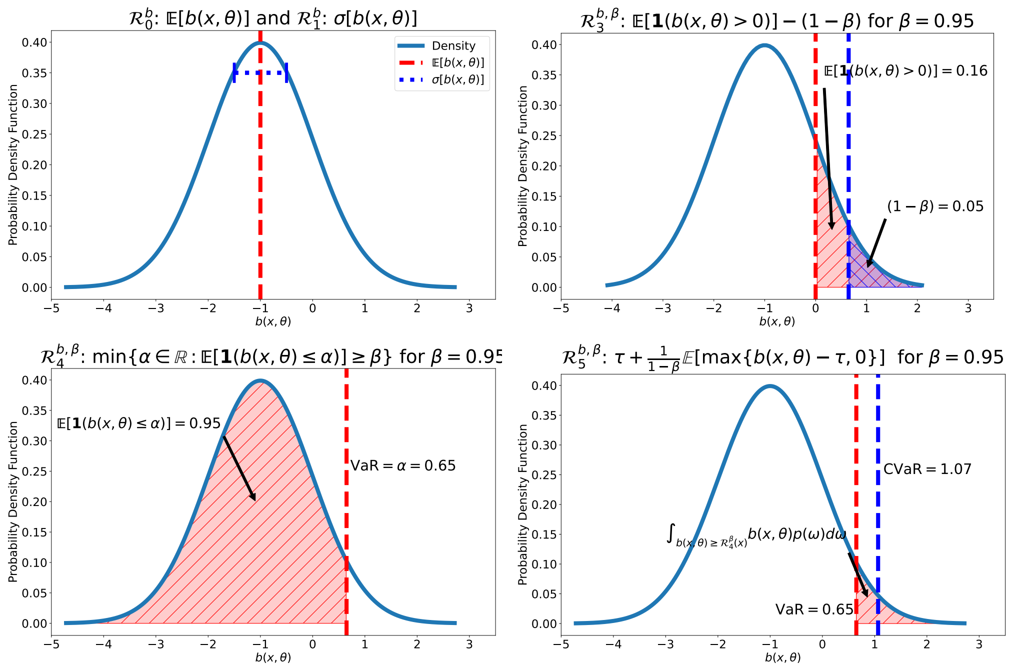


Figure 9.1: Examples of $\mathcal{R}_0^{b,\beta}$, $\mathcal{R}_1^{b,\beta}$ (both top left), $\mathcal{R}_3^{b,\beta}$ (top right), $\mathcal{R}_4^{b,\beta}$ (bottom left) and $\mathcal{R}_5^{b,\beta}$ (bottom right) for $\beta = 0.95$ for a one-dimensional Gaussian distribution $\theta \sim \mathcal{N}(-1, 1)$.

9.2 A brief review of existing work in stochastic optimization

In the this section, we provide a brief historical context and survey of the various optimization techniques employed to solve Eq. (9.1). We also refer to a recent review article [192]—already cited in the context of deterministic optimization methods—which offers an extensive overview of stochastic derivative-free optimization methods and our own review in [221].

One optimization approach is the Sample Average Approximation (SAA) [180]. This method employs a predetermined set of samples, $\{\boldsymbol{\theta}_i\}_{i=1}^N$, to approximate the robustness measures as $R_N \approx \mathcal{R}$. Throughout the optimization process, this sample set remains consistent to minimize the sample approximated objective function $\mathcal{R}^f$. This methodology results in approximate solutions of Eq. (9.1), dependent on the specific sample choice. To mitigate the approximation error associated with this method, multiple optimization runs are usually averaged, or the sample size is increased. Refer to [7, 277, 285] for more detail. An error analysis of SAA for constrained optimization problems is available in [32]. The advantage of SAA is its ability to eliminate the noise induced by varying sample approximations between optimization steps, allowing deterministic black-box optimization methods to solve the optimization problem.

Other strategies involve generating new samples from the uncertain parameter $\boldsymbol{\theta}$ each time the robustness measures are evaluated. Owing to the re-sampling process, evaluations of the approximate robustness measures $R_N(x)$ exhibit 'sampling noise'. As a result, solving Eq. (9.1) necessitates the use of stochastic optimization methods. If the noise is minimal, as might be the case with a sufficiently large sample size N, pattern-search methods, as we have already presented in Section 8.2, can be applied to solve the optimization problem. These methods are less susceptible to noise in the evaluations of the robust objective and constraints because they avoid gradient approximations. Since the pioneering works of Hooke and Jeeves [160], and Nelder and Mead [237, 297], there has been considerable research on expanding and refining effective direct-search optimization techniques [15, 18, 17, 98, 210, 211, 261, 262], which we have discussed in the previous chapter. More recent extensions are also available, specifically for stochastic problems for the unconstrained [20] and constrained case [104], where using a progressive barrier allows to stay feasible.

Optimization based on surrogate models [23, 52, 77, 175, 217, 270, 271, 278] is another class of methods employed to solve Eq. (9.1). These methods can prove convergence, given that the gradient approximations are sufficiently accurate; see [61, 67, 68, 154, 191]. Here, 'sufficiently accurate' implies that the gradient approximation needs to improve in accuracy as it nears an optimal solution. This concept is included in the derivative-free stochastic optimization procedures STRONG [64] and ASTRO-DF [286]. These procedures diminish the noise in black-box evaluations by taking averages over an increasing number of samples as they approach an optimal design.

So far, we have discussed optimization methods that necessitate a reduction in the magnitude of noise in the measure approximations. We now shift our focus to methods that do not require this. In 1951, the field was revolutionized when Robbins and Monroe [274] introduced the Stochastic Approximation (SA) method. Since then, SA has been adapted to a range of gradient approximation techniques, such as those proposed by Kiefer and Wolfowitz (KWSA) [179], and Spall [295, 296, 315] in the form of Simultaneous Perturbation Stochastic Approximation (SPSA). For an in-depth introduction and theoretical analysis of SA methods, we refer to [38, 177, 190]. It is worth noting that all SA methods involve the careful selection of several technical parameters, such as step and stencil sizes. Despite the extensive literature and theoretical results available, this selection process remains a daunting task in the application of SA approaches. While both optimal and heuristic choices exist [296], they are highly problem-dependent and have a significant impact on the performance and efficiency of SA methods. With the rapid growth of machine learning as the primary application field, variations of the stochastic approximation method continue to evolve, particularly in the realm of stochastic gradient descent. We refer to [54] for a recent review of these developments.

Bayesian Global Optimization (BGO) [228, 229] is yet another approach that can be employed to solve the problem in Eq. (9.1). In BGO, a GP is utilized to approximate the objective function. Using the GP surrogate and its Gaussian properties, metrics like the expected improvement or knowledge gradients are defined, which are used to globally find the next best optimization step. This method has been extensively discussed in studies like [124, 170]. Handling nonlinear constraints within BGO has only recently begun to draw attention in literature [146]. One specific method, known as constrained Bayesian Optimization (cBO), utilizes expected constrained improvement optimization, as illustrated in [129]. For further developments in this field, we refer to more recent studies such as [113] and [200].

Looking at the literature, we see that there is increased interest in the field of OUU in recent years with new advances both in optimization as well as in UQ. Our second contribution of this thesis, the derivative-free optimization method SNOWPAC builds on literature presented here. It uses a trust-region approach and sampling estimators to estimate the measures from Section 9.1. It also employs GPs but uses them as second surrogates to reduce noise instead of a direct surrogate for the objective, differently to BGO. We will present our own derivative-free optimization method SNOWPAC in Part IV.

10 Dakota

Dakota (Design Analysis Kit for Optimization and Terascale Applications) [4, 91] is a powerful, flexible, open-source software toolkit that provides a cohesive interface for describing and solving optimization and UQ problems. The toolkit is developed at Sandia National Laboratories and is primarily used for parameter optimization, UQ, parameter estimation, sensitivity analysis, and design of experiments, amongst other related tasks. It supports a range of optimization algorithms, including derivative-based methods, derivative-free methods, and global optimization methods.

Dakota's functionality is broadly split into three parts: an interface that connects Dakota to the user's application code, a range of algorithms for optimization and UQ, and an input/output system for handling communication between these components. This flexible design allows Dakota to be easily integrated with various existing codes and software systems, making it a popular choice for researchers and scientists.

In these regards, Dakota not only offers its own methods, but also functions as the driver and coupler between methods and applications. In UQ, Dakota, e.g., takes care of sampling random variables or integrating statistics while communicating the input to the application and routing the results back to the method. In optimization, Dakota propagates the current design to the solver for evaluation and manages the result and the optimization process.

The software is also designed to be highly scalable. Dakota provides robust support for parallel computing, making it a suitable tool for handling large-scale, computationally demanding applications. It is commonly used across a variety of fields, including engineering, scientific research, and operations research. Written in C++, Dakota is open source under GNU LGPL, finds utility in a wide range of fields including defense programs for the US Department of Energy and US Department of Defense [134], climate modeling [125], nuclear power [316, 335], renewable energy [162, 265], and many others.

Dakota interacts with the user via an input file which steers the program. Using the parameter `environment`, the user can denote where the results should be written. The field `method` describes the method that is used and also includes all its settings. The `model` points to the different parameters describing it. In `variables`, the user sets all the variables required. Here, e.g., specific variables for the application or uncertain variables for the method can be defined. The field `interface` denotes the link to the application and offers options for parallelization. Finally, in the option `responses`, the user can denote the number of response functions and if gradients are computed.

Example 6 (Dakota input file)**.** *We give an example of an input file for MC sampling on the one-dimensional diffusion equation in Listing 10.1. The example uses 1000 MC samples, with a fixed seed and has seven uniform random variables in* $[-1, 1]$*. It* *furthermore defines two problem specific integer variables for the number of grid coordinates and Fourier modes of the transient diffusion equation. The samples are picked randomly, whereas deterministic sampling methods (such as QMC) are also available. The transient diffusion equation is directly implemented in Dakota but also linking to external applications is possible. We can further specify if gradients or Hessians should be computed.*

```
environment,
    tabular_data

method,
   sampling
      sample_type random
      samples = 1000

model,
    id_model = 'HF'
    variables_pointer = 'HF_VARS'
    simulation

variables,
    id_variables = 'HF_VARS'
    uniform_uncertain = 7
      lower_bounds     = 7*-1.
      upper_bounds     = 7* 1.
    discrete_state_set
      integer = 2
        num_set_values = 1 1
        set_values = 200 # number of spatial coords
                     21  # number of Fourier solution modes
        initial_state = 200 21
        descriptors 'N_x' 'N_mod'

interface,
     direct
      analysis_driver = 'transient_diffusion_1d'

responses,
    response_functions = 1
    no_gradients
    no_hessians
```

Listing 10.1: Example Dakota input file for MC simulation for one-dimensional diffusion equation.

Dakota is the main software framework, where our contributions are implemented in or linked by. Our first contribution of this work on MLMC estimators for higher-order moments, presented in Part III, is implemented directly in Dakota. Our second contribution, the derivative-free stochastic optimization method SNOWPAC, presented in Part IV, can be employed as an external solver. The coupling of both contributions, which we present in Part V, is then orchestrated by Dakota. It offers offers a flexible and adjustable framework, integrating all our parts and linking it to the black-box problem that we want to solve.

11 Gap analysis

The motivation for this work originates from the need to optimize challenging black-box OUU problems computationally efficiently (and accurately). This not only requires a performant optimization method, but also capable sampling estimators for the involved statistics. In both areas, OUU and sampling, we perceived a gap in literature for our requirements, which we are going to analyze more detail in this chapter.

MLMC methods were introduced for estimating the expected value. Moreover, most extensions developed for the approach focus on the mean. In our field of application of OUU, we, however, are interested in higher order moments like the variance or measures like the standard deviation or scalarization. MLMC estimators for these measures are not available in literature.

Thus, we identified this first gap to be able to efficiently apply MLMC methods in OUU. While first developments in this area were done by Bierig et al. [40], their work focused on unbiased estimators for the variance only. The work by Krumscheid et al. [188, 189] also regarded general higher order central moments. Ganesh et al. [127] and Ayoul-Guilmard et al. [25], were the first to look at measures and the CVaR specifically. However, to the best of our knowledge, we are the first to look at the standard deviation and scalarization. This requires not only the derivation of the estimators themselves but also their estimator variances. We present the estimators and the corresponding optimal resource allocation problem as our first major contribution in Part III.

In reviewing the body of existing optimization methods, we have identified another significant gap: a lack of methods that effectively combine stochastic optimization problems under nonlinear constraints with derivative-free approaches. These three components represent critical aspects of many practical optimization scenarios, especially in OUU for computational challenging black-box problems. Most existing methodologies tend to focus on one or two aspects out of the three, but not on all three at the same time. In black-box OUU problems, we, on the one hand, cannot expect to have access to gradients, but, on the other hand, can expect stochastic conditions while the problem also often comes with challenging constraints. The limited attention being paid to the intersection of these properties has left a noticeable void in the collection of optimization methods.

In the Venn diagram in Figure 11.1, we present literature that employs the different components or combinations thereof. Here, we only show an extract of literature in the different fields. However, our contribution, marked in bold and published in [221], is one of only two publications for the center intersection of all three components. As far as we are aware, the only other work has been published by Dzahini et al. [104], which employs a direct-search method with a progressive barrier approach.

Bringing all three components together greatly generalizes the applicability of a method to a variety of problems. We do not require different methods if gradients are not available, if the optimization problem is constrained, or if the problem is deterministic. This is what we require for black-box OUU problems. In Part IV, we address this gap by presenting SNOWPAC, a method that is equipped to handle stochastic optimization problems under nonlinear constraints using a derivative-free method.

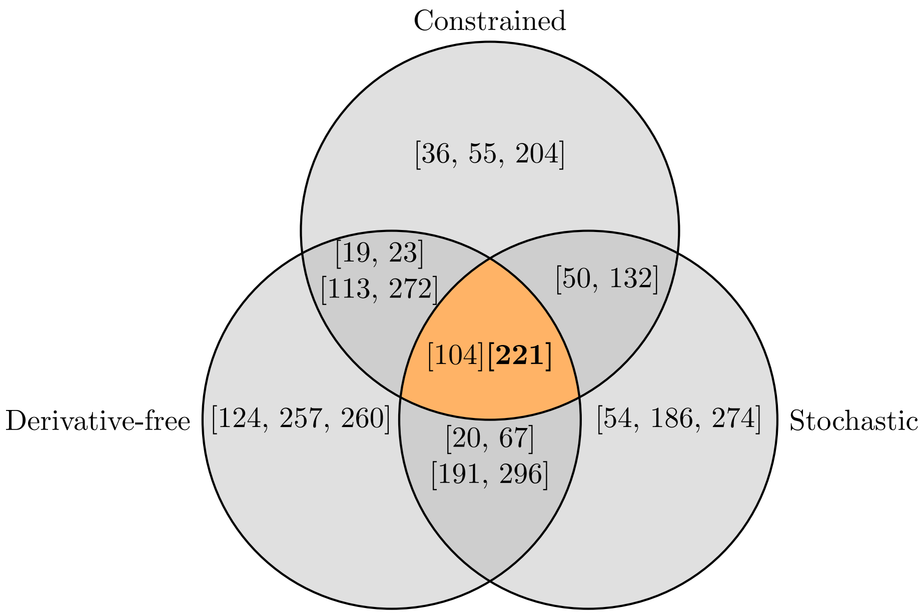


Figure 11.1: Venn diagram about the different classes of optimization methods showing literature in the fields.

Finally, by combining both contributions together in a software framework, we are able to fill the gap of being able to tackle computationally challenging black-box problems in OUU. To our knowledge, we are the first to look at problems of this kind, optimizing black-box OUU problems with a multilevel derivative-free approach. We present this contribution in Part V.

Part III

Multilevel Monte Carlo for higher-order moments

It's the questions we can't answer that teach us the most. They teach us how to think. If you give a man an answer, all he gains is a little fact. But give him a question and he'll look for his own answers.
—*Patrick Rothfuss [276]*

12 Multilevel estimators for higher-order moments

In Sections 6.3 and 6.4, we discussed single level MC estimator and the MLMC method for the mean. We presented the resource allocation problem in Eq. (6.26), where we optimize for the optimal sample allocation, $\overset{*}{\mathbf{N}}^{\mathbb{E}}$, targeting a certain variance of the multilevel estimator $\mathbb{V}[\widehat{\mu}_{1,\mathrm{ML}}] = \epsilon_{\mathbb{E}}^2$. We can generalize this problem for higher-order statistics as

$$\begin{aligned} \overset{*}{\mathbf{N}}^{\mathbb{X}} &= \underset{\mathbf{N}^{\mathbb{X}}}{\arg\min}\, C_T^{\mathbb{X}}, \\ \text{s.t. } \mathbb{V}\left[\hat{X}_{\mathrm{ML}}\right] &= \epsilon_{\mathbb{X}}^2, \end{aligned} \tag{12.1}$$

where $\hat{X}_{\mathrm{ML}} \approx \mathbb{X}$ is the corresponding multilevel estimator for the SoI $\mathbb{X}$, e.g. $\hat{X}_{\mathrm{ML}} = \widehat{\mu}_{1,\mathrm{ML}}$ for $\mathbb{X} = \mathbb{E}$ and $C_T^{\mathbb{X}} := \sum_{\ell=1}^{L} C_\ell N_\ell^{\mathbb{X}}$ is the total computational cost of the estimator.

In the upcoming chapter, we are going to introduce our first major contribution of this thesis, namely higher-order estimators for MLMC and their optimal resource allocation. We focus on new formulations for variance, standard deviation and a linear combination of mean and standard deviation. These are all commonly used statistics in OUU, as we have presented in Chap. 9. The main task will be to find a formulation for the variance of the multilevel estimators such that we can solve Eq. (12.1).

Our first minor contributions concerns an unbiased single level estimators for the fourth moment and for the RMSE of the variance in Section 12.1, which we will need. We then move to the main contribution, the new MLMC estimators in Sections 12.2-12.4, which is based on work in [223] and [224]. We will continue comparing the new estimators to other work in the field in Section 12.5. Afterwards, we will present numerical results in Chap. 13.

12.1 Higher-order MC estimators

Besides the expected value, MC sampling can also be used as unbiased estimator for the variance, i.e. the second centered moment, as

$$\widehat{\mu}_2 = \frac{1}{N-1} \sum_{i=1}^{N} (Q^{(i)} - \widehat{\mu}_1)^2. \tag{12.2}$$

It is a well-known result that this estimator, thanks to the use of the Bessel correction, is also unbiased.

Lemma 7. *The variance estimator $\widehat{\mu}_2$ is unbiased, i.e.*

$$\mathbb{E}[\widehat{\mu}_2] = \mu_2 = \mathbb{V}[Q] \tag{12.3}$$

Since the result is well-known and the proof is technical, it can be found in App. C.1.

Thus, we again can compute its RMSE from its variance, which has a closed-form expression given in [230]:

$$\mathbb{V}[\widehat{\mu}_2] = \frac{1}{N}(\mu_4 - \frac{N-3}{N-1}\mu_2^2). \tag{12.4}$$

However, the variance depends on both the exact statistics of the squared second and fourth central moments of the QoI, μ_2^2 and μ_4, respectively. Estimators for both statistics are by default biased.

We derive an unbiased estimator for Eq. (12.4) when we use unbiased estimators for μ_2 and μ_4 in the following Lemma. This result is also given in [189], though derived by using h-statistics.

Lemma 8. *Let $\widehat{\mu}_2$ and $\widehat{\mu}_4$ be unbiased estimators for the second and fourth central moment. The unbiased estimator of the variance of the second central moment is then given as*

$$\mathbb{V}[\widehat{\mu}_2] \approx \widehat{\mu}_2[\widehat{\mu}_2] = \frac{(N-1)}{N^2-2N+3}\left(\widehat{\mu}_4 - \frac{N-3}{N-1}\widehat{\mu_2^2}\right). \tag{12.5}$$

See C.2 for the proof.

Indeed, in Eq. (12.4), we can estimate the variance $\mathbb{V}[\widehat{\mu}_2]$ by relying on sample estimators for both the second μ_2 and fourth μ_4 central moments. Since an unbiased estimator for the variance is already available (see Eq. (12.2)), we only need to obtain an unbiased estimator $\widehat{\mu}_4$ for the fourth central moment μ_4. Obtaining this unbiased estimator, from its biased counterpart, is discussed in the following lemma.

Lemma 9. *Let $\widehat{\mu}_{4,biased} = \frac{1}{N}\sum_{i=1}^{N}(Q^{(i)} - \widehat{\mu}_1)^4$ be a biased estimator for the fourth central moment and let $\widehat{\mu}_2$ be an unbiased estimator for the second central moment as given in* (12.2)*. Then, an unbiased estimator for the fourth central moment is given as*

$$\widehat{\mu}_4 = \frac{1}{(N^2-3N+3) - \frac{(6N-9)(N^2-N)}{N(N^2-2N+3)}}\left(\frac{N^3}{N-1}\widehat{\mu}_{4,biased} - \frac{(6N-9)(N^2-N)}{N^2-2N+3}\widehat{\mu_2^2}\right). \tag{12.6}$$

See C.3 for the proof.

Finally, the last single fidelity estimator we need to discuss is the standard deviation, which can be approximated directly from the variance estimator as

$$\widehat{\sigma}_{\text{biased}} = \sqrt{\widehat{\mu}_2}. \tag{12.7}$$

This latter case introduces a number of challenges. First, an unbiased version of the estimator cannot be easily derived (even if we rely on the unbiased variance from

Eq. (12.2)), mainly due to the square root operator. Second, and as a consequence, the variance of this estimator cannot be obtained in closed-form either. To overcome this difficulty and get an expression to use for resource allocation purposes, we can rely on the *Delta method* [209]. It employs a Taylor series expansion to find the approximate probability distribution for a function of an asymptotically normal distributed estimator, which, in our case, will be the square root function and the variance estimator respectively:

Lemma 10. *Let us assume that $\widehat{\mu}_2$ is asymptotically normal distributed and that mean and variance exist. The variance of $\widehat{\sigma}_{biased}$ can be approximated by using the Delta method [209] as*

$$\mathbb{V}[\widehat{\sigma}_{biased}] \approx \frac{1}{4\widehat{\mu}_2}\mathbb{V}[\widehat{\mu}_2]. \tag{12.8}$$

See C.4 for the proof.

We again point out that this estimator for the variance, $\widehat{\mu}_2$, does not necessarily follow a normal distribution, resulting in an approximation.

12.2 Multilevel sample allocation for the variance estimator

In general, for our OUU problems we are not only interested in the expected value but also in higher-order moments, as explained in Section 9.1. The standard deviation plays an important role specifically in the field of OUU, e.g., in the measures $\mathcal{R}_1^b$ and $\mathcal{R}_2^b$ as mentioned in Chap. 9. As noted, when optimizing over the standard deviation, we decrease the variation in our optimal design.

To find a MLMC estimator for the standard deviation, we have a look at the variance first. This is our first contribution in finding the optimal resource allocation for higher-order moments. Here, we build on the MLMC estimator for the mean, as outlined in Section 6.4.

Let us start by defining the MLMC estimator for the variance as follows:

$$\begin{aligned}\mathbb{V}[Q_L] \approx \widehat{\mu}_{2,\mathrm{ML}}[Q_L] &:= \sum_{\ell=1}^{L} \widehat{\mu}_2[Q_\ell] - \widehat{\mu}_2[Q_{\ell-1}] \\ &= \sum_{\ell=1}^{L} (\widehat{\mu}_{2,\ell} - \widehat{\mu}_{2,\ell-1}) \\ &= \sum_{\ell=1}^{L} \frac{1}{N_\ell - 1}\left(\sum_{i=1}^{N_\ell} (Q_\ell^{(i)} - \widehat{\mu}_{1,\ell})^2 - (Q_{\ell-1}^{(i)} - \widehat{\mu}_{1,\ell-1})^2 \right).\end{aligned} \tag{12.9}$$

Similarly to the mean, we use a telescopic sum over levels to introduce the level differences and substitute the single-level sampling estimator for each term. The estimator $\widehat{\mu}_{2,\mathrm{ML}}$ is unbiased, since we use unbiased estimators for the variance on every level.

To find the optimal resource alloction $\overset{*}{\mathbf{N}}^{\mathbb{V}}$, we have to solve the following resource allocation problem:

$$\begin{aligned}\overset{*}{\mathbf{N}}^{\mathbb{V}} &= \arg\min_{\mathbf{N}^{\mathbb{V}}} C_T^{\mathbb{V}}, \\ \text{s.t. } \mathbb{V}[\widehat{\mu}_{2,\mathrm{ML}}] &= \epsilon_{\mathbb{V}}^2,\end{aligned} \tag{12.10}$$

with $C_T^{\mathbb{V}} := \sum_{\ell=1}^{L} C_\ell N_\ell^{\mathbb{V}}$. Note the small but significant difference to the resource allocation problem for the mean in Eq. (6.26): instead of using the variance of the mean estimator in the constraint, we now constrain on the variance of the variance estimator for a target accuracy $\epsilon_{\mathbb{V}}^2$. Hence, we *target the variance* instead of the mean.

For the solution of Eq. (12.10), we need the variance of the MLMC variance estimator $\widehat{\mu}_{2,\mathrm{ML}}$:

$$\begin{aligned}\mathbb{V}[\widehat{\mu}_{2,\mathrm{ML}}] = \mathbb{V}\left[\sum_{\ell=1}^{L}(\widehat{\mu}_{2,\ell} - \widehat{\mu}_{2,\ell-1})\right] &= \sum_{\ell=1}^{L} \mathbb{V}\left[\widehat{\mu}_{2,\ell} - \widehat{\mu}_{2,\ell-1}\right] \\ &= \sum_{\ell=1}^{L} \mathbb{V}[\widehat{\mu}_{2,\ell}] + \mathbb{V}[\widehat{\mu}_{2,\ell-1}] - 2\mathbb{C}\mathrm{ov}[\widehat{\mu}_{2,\ell}, \widehat{\mu}_{2,\ell-1}].\end{aligned} \tag{12.11}$$

As for the mean, this equation depends on the assumption of inter-level independence and intra-level dependence of samples. If clear from context, we use the notation $\widehat{\mu}_{2,\ell-1} = \widehat{\mu}_{2,\ell-1}\left[Q_{\ell-1}[\mathbf{x}, \boldsymbol{\theta}_\ell]\right]$, wherein the QoI is evaluated on level $\ell - 1$, using identical samples $\boldsymbol{\theta}_\ell$ from level ℓ. Thus, while having independence over levels, we have a dependence between terms on the same level. The ensuing covariance term represents this dependence as well, utilizing the same compact notation: $\mathbb{C}\mathrm{ov}[\widehat{\mu}_{2,\ell}, \widehat{\mu}_{2,\ell-1}] = \mathbb{C}\mathrm{ov}\left[\widehat{\mu}_{2,\ell}\left[Q_\ell[\mathbf{x}, \boldsymbol{\theta}_\ell]\right], \widehat{\mu}_{2,\ell-1}\left[Q_{\ell-1}[\mathbf{x}, \boldsymbol{\theta}_\ell]\right]\right]$.

In Section 6.3, we have examined the estimation of single-fidelity variance expressions for the terms $\mathbb{V}[\widehat{\mu}_{2,\ell}]$ and $\mathbb{V}[\widehat{\mu}_{2,\ell-1}]$. We refer to Eq. (12.4) and its unbiased estimator in Eq. (12.5). However, in Eq. (12.11), an extra term, $\mathbb{C}\mathrm{ov}[\widehat{\mu}_{2,\ell}, \widehat{\mu}_{2,\ell-1}]$, requires evaluation. We will provide the expression for this term in the succeeding lemma.

Lemma 11. *Let $\widehat{\mu}_{2,\ell}$ and $\widehat{\mu}_{2,\ell-1}$ be unbiased single level estimators for the respective level ℓ and $\ell - 1$ as described in Eq.* (12.2)*. Then, the covariance term in Eq.* (12.11) *is given as*

$$\begin{aligned}\mathbb{C}ov[\widehat{\mu}_{2,\ell}, \widehat{\mu}_{2,\ell-1}] &= \frac{1}{N_\ell}\mathbb{E}[\widehat{\mu}_{2,\ell}\widehat{\mu}_{2,\ell-1}] \\ &+ \frac{1}{N_\ell(N_\ell - 1)}\left(\mathbb{E}[Q_\ell Q_{\ell-1}]^2 - 2\mathbb{E}[Q_\ell Q_{\ell-1}]\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}] + (\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}])^2\right)\end{aligned} \tag{12.12}$$

where

$$\begin{aligned}\mathbb{E}[\widehat{\mu}_{2,\ell}\widehat{\mu}_{2,\ell-1}] = &\ \mathbb{E}[{Q_\ell}^2 {Q_{\ell-1}}^2] - \mathbb{E}[{Q_\ell}^2]\mathbb{E}[{Q_{\ell-1}}^2] \\ &- 2\mathbb{E}[Q_{\ell-1}]\mathbb{E}[{Q_\ell}^2 Q_{\ell-1}] + 2\mathbb{E}[Q_{\ell-1}]^2\mathbb{E}[{Q_\ell}^2] \\ &- 2\mathbb{E}[Q_\ell]\mathbb{E}[Q_\ell {Q_{\ell-1}}^2] + 2\mathbb{E}[Q_\ell]^2\mathbb{E}[{Q_{\ell-1}}^2] \\ &+ 4\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}]\mathbb{E}[Q_\ell Q_{\ell-1}] - 4\mathbb{E}[Q_\ell]^2\mathbb{E}[Q_{\ell-1}]^2.\end{aligned} \tag{12.13}$$

See C.5 for the proof.

It is important to point out that the product of expected values can yield biased estimators, even when each expected value is independently approximated by unbiased estimators. This is specifically the case for the expectations in Eq. (12.12) and Eq. (12.13). Consequently, we proceed to formulate unbiased estimators for the double, triple, and quadruple products of expected values in the following lemmas:

Lemma 12. *Let* $(\widehat{\mu}_{1,\ell}\widehat{\mu}_{1,\ell-1})_{biased} = \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_\ell^{(i)} \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_{\ell-1}^{(i)}$ *be a biased estimator for the product of expected value estimators. Then, an unbiased estimator is given as*

$$\widehat{\mu}_{1,\ell}\widehat{\mu}_{1,\ell-1} = \frac{N_\ell}{N_\ell - 1}(\widehat{\mu}_{1,\ell}\widehat{\mu}_{1,\ell-1})_{biased} - \frac{1}{N_\ell - 1}\widehat{\mu}_{1,\ell}[Q_\ell Q_{\ell-1}] \tag{12.14}$$

See C.6 for the proof.

Lemma 13. *Let* $(\widehat{\mu}_{1,\ell_1}\widehat{\mu}_{1,\ell_2}\widehat{\mu}_{1,\ell_3})_{biased} = \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_{\ell_1}^{(i)} \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_{\ell_2}^{(i)} \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_{\ell_3}^{(i)}$ *be a biased estimator for the triple products of expected value estimators. Additionally, assume an unbiased product of mean estimators based on Eq.* (12.14). *Then, an unbiased estimator is given as*

$$\begin{aligned}\widehat{\mu}_{1,\ell_1}\widehat{\mu}_{1,\ell_2}\widehat{\mu}_{1,\ell_3} &= \frac{N_\ell^2}{(N_\ell - 1)(N_\ell - 2)}(\widehat{\mu}_{1,\ell_1}\widehat{\mu}_{1,\ell_2}\widehat{\mu}_{1,\ell_3})_{biased} \\ &- \frac{1}{N_\ell - 2}\Big(\widehat{\mu}_{1,\ell}[Q_{\ell_1}Q_{\ell_2}]\widehat{\mu}_{1,\ell}[Q_{\ell_3}] + \widehat{\mu}_{1,\ell}[Q_{\ell_1}Q_{\ell_3}]\widehat{\mu}_{1,\ell}[Q_{\ell_2}] \\ &+ \widehat{\mu}_{1,\ell}[Q_{\ell_2}Q_{\ell_3}]\widehat{\mu}_{1,\ell}[Q_{\ell_1}]\Big) - \frac{1}{(N_\ell - 1)(N_\ell - 2)}\widehat{\mu}_{1,\ell}[Q_{\ell_1}Q_{\ell_2}Q_{\ell_3}]\end{aligned} \tag{12.15}$$

See C.7 for the proof.

Lemma 14. *Let* $(\widehat{\mu}_{1,\ell}^2\widehat{\mu}_{1,\ell-1}^2)_{biased} = \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_{\ell_1} \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_{\ell_1} \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_{\ell-1}^{(i)} \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_{\ell-1}^{(i)}$ *be a biased estimator. Additionally, assume an unbiased estimator for double and triple product of mean estimators is given based on Eq.* (12.14) *and Eq.* (12.15), *respectively. Then, an unbiased estimator for* $\widehat{\mu}_{1,\ell}^2\widehat{\mu}_{1,\ell-1}^2$ *is given as*

$$\begin{aligned}\widehat{\mu}_{1,\ell}^2\widehat{\mu}_{1,\ell-1}^2 &= \frac{N_\ell^3}{(N_\ell - 1)(N_\ell - 2)(N_\ell - 3)}(\widehat{\mu}_{1,\ell}^2\widehat{\mu}_{1,\ell-1}^2)_{biased} \\ &- \frac{1}{N_\ell - 3}\Big(\widehat{\mu}_{1,\ell}[{Q_\ell}^2]\widehat{\mu}_{1,\ell}[Q_{\ell-1}]^2 + 4\widehat{\mu}_{1,\ell}[Q_\ell Q_{\ell-1}]\widehat{\mu}_{1,\ell}[Q_\ell]\widehat{\mu}_{1,\ell}[Q_{\ell-1}] \\ &+ \widehat{\mu}_{1,\ell}[Q_\ell]^2\widehat{\mu}_{1,\ell}[{Q_{\ell-1}}^2]\Big) \\ &- \frac{1}{(N_\ell - 2)(N_\ell - 3)}\Big(\widehat{\mu}_{1,\ell}[{Q_\ell}^2]\widehat{\mu}_{1,\ell}[{Q_{\ell-1}}^2] + 2\widehat{\mu}_{1,\ell}[Q_\ell Q_{\ell-1}]^2 \\ &+ 2\widehat{\mu}_{1,\ell}[{Q_\ell}^2 Q_{\ell-1}]\widehat{\mu}_{1,\ell}[Q_{\ell-1}] + 2\widehat{\mu}_{1,\ell}[Q_\ell]\widehat{\mu}_{1,\ell}[Q_\ell {Q_{\ell-1}}^2]\Big) \\ &- \frac{1}{(N_\ell - 1)(N_\ell - 2)(N_\ell - 3)}\widehat{\mu}_{1,\ell}[{Q_\ell}^2 {Q_{\ell-1}}^2].\end{aligned} \tag{12.16}$$

See C.8 for the proof.

Having derived these unbiased estimators for the double, triple, and quadruple product of expected values, we are now moving forward to compute an unbiased estimator for the covariance. This is achieved by leveraging the property of linearity of the expected value.

Lemma 15. *Let $\widehat{\mu}_{2,\ell}$ and $\widehat{\mu}_{2,\ell-1}$ be unbiased single level estimators for the respective level ℓ and $\ell-1$ as described in* (12.2)*. Additionally, let $\widehat{\mu}_1$ be unbiased estimators for the respective expected value as described in* (6.16)*. An unbiased estimator for the covariance term in Lemma 11 is given as*

$$\begin{aligned}\widehat{\mathbb{C}ov}[\widehat{\mu}_{2,\ell},\widehat{\mu}_{2,\ell-1}] \approx{}& \frac{1}{N_\ell}\widehat{\mu}_1[\widehat{\mu}_{2,\ell}\widehat{\mu}_{2,\ell-1}] \\ &+ \frac{1}{N_\ell(N_\ell-1)}\Big(\widehat{\mu}_{1,\ell}[Q_\ell Q_{\ell-1}] - 2\widehat{\mu}_{1,\ell}[Q_\ell Q_{\ell-1}]\widehat{\mu}_{1,\ell}\widehat{\mu}_{1,\ell-1} - (\widehat{\mu}_{1,\ell}\widehat{\mu}_{1,\ell-1})^2\Big)\end{aligned} \tag{12.17}$$

where

$$\begin{aligned}\widehat{\mu}_1[\widehat{\mu}_{2,\ell}\widehat{\mu}_{2,\ell-1}] ={}& \widehat{\mu}_{1,\ell}\left[{Q_\ell}^2{Q_{\ell-1}}^2\right] - 2\widehat{\mu}_{1,\ell}\left[{Q_\ell}^2 Q_{\ell-1}\right]\widehat{\mu}_{1,\ell}\left[Q_{\ell-1}\right] \\ &+ 2\widehat{\mu}_{1,\ell}\left[Q_{\ell-1}\right]^2\widehat{\mu}_{1,\ell}\left[{Q_\ell}^2\right] - 2\widehat{\mu}_{1,\ell}\left[Q_\ell\right]\widehat{\mu}_{1,\ell}\left[Q_\ell {Q_{\ell-1}}^2\right] \\ &+ 4\widehat{\mu}_{1,\ell}\left[Q_{\ell-1}\right]\widehat{\mu}_{1,\ell}\left[Q_\ell\right]\widehat{\mu}_{1,\ell}\left[Q_\ell Q_{\ell-1}\right] + 2\widehat{\mu}_{1,\ell}\left[Q_\ell\right]^2\widehat{\mu}_{1,\ell}\left[{Q_{\ell-1}}^2\right] \\ &- 4\widehat{\mu}_{1,\ell}\left[Q_\ell\right]^2\widehat{\mu}_{1,\ell}\left[Q_{\ell-1}\right]^2 - \widehat{\mu}_{1,\ell}\left[{Q_\ell}^2\right]\widehat{\mu}_{1,\ell}\left[{Q_{\ell-1}}^2\right].\end{aligned} \tag{12.18}$$

See C.9 for the proof.

By integrating all the components of the covariance approximation from Eq. (12.17) with the unbiased variance estimator from Eq.(12.5), we finally arrive at an unbiased estimator for Eq. (12.11).

Lemma 16. *The estimator*

$$\widehat{\mu}_2[\widehat{\mu}_{2,ML}] = \sum_{\ell=1}^{L}\widehat{\mu}_2[\widehat{\mu}_{2,\ell}] + \widehat{\mu}_2[\widehat{\mu}_{2,\ell-1}] - 2\widehat{\mathbb{C}ov}[\widehat{\mu}_{2,\ell},\widehat{\mu}_{2,\ell-1}] \tag{12.19}$$

is an unbiased estimator for

$$\mathbb{V}[\widehat{\mu}_{2,ML}] = \sum_{\ell=1}^{L}\mathbb{V}[\widehat{\mu}_{2,\ell}] + \mathbb{V}[\widehat{\mu}_{2,\ell-1}] - 2\mathbb{C}ov[\widehat{\mu}_{2,\ell},\widehat{\mu}_{2,\ell-1}]. \tag{12.20}$$

See C.10 for the proof.

Having derived an unbiased estimator for the variance of the MLMC variance estimator, we can solve the resource allocation problem from Eq. (12.10). We find the optimal resource allocation $\overset{*}{\mathbf{N}}^{\mathbb{V}}$ given a certain accuracy $\epsilon^2_{\mathbb{V}}$ for our MLMC variance estimator. Eq. (12.10) does not offer an analytic solution. This is due to higher-order terms of $N_\ell^{\mathbb{V}}$ showing up in the estimator, which make solving the resource allocation problem non-trivial. Approaches for an analytic approximation, which we will discuss in Section 12.5, exist; nevertheless, we can also resort to numerical optimization to find the optimal sample allocation. Here, we use the exact formulation of the estimators and their gradients, solving the equality-constrained problem of Eq. (12.10) numerically.

12.3 Multilevel sample allocation for the standard deviation estimator

Next, we move to presenting a MLMC estimator for the standard deviation. We use it to compute the measures $\mathcal{R}_1^b$ and $\mathcal{R}_2^b$ as mentioned in Chap. 9. Capitalizing on the previous findings and the relation between the standard deviation and variance, we employ the subsequent MLMC estimator

$$\widehat{\sigma}_{\mathrm{ML,biased}} := \sqrt{\widehat{\mu}_{2,\mathrm{ML}}}. \tag{12.21}$$

As we point out in the subscript, this estimator is biased.

Again, we adapt the resource allocation accordingly, where we now require the variance of the standard deviation estimator

$$\begin{aligned} \overset{*}{\mathbf{N}}{}^{\sigma} &= \underset{\mathbf{N}^{\sigma}}{\arg\min}\, C_T^{\sigma}, \\ \text{s.t. } \mathbb{V}[\widehat{\sigma}_{\mathrm{ML,biased}}] &= \epsilon_{\sigma}^2, \end{aligned} \tag{12.22}$$

minimizing the total cost $C_T^{\sigma} := \sum_{\ell=1}^{L} C_\ell N_\ell^{\sigma}$. We call this resource allocation problem *targeting the standard deviation.*

Despite the quick derivation for the multilevel estimator of the standard deviation, the derivation of its variance is complex, mainly due to the involved square root. To approximate its variance, we once again resort to the Delta method. This method, assuming a normal distribution for the underlying estimator, facilitates the following expression:

$$\mathbb{V}[\widehat{\sigma}_{\mathrm{ML,biased}}] \approx \frac{1}{4\widehat{\mu}_2^{\mathrm{ML}}} \mathbb{V}[\widehat{\mu}_{2,\mathrm{ML}}], \tag{12.23}$$

This approach is akin to what we have discussed in Section 6.3 concerning its single-fidelity expression. Likewise, in this case, solving the resource allocation problem, as presented in Eq. (12.22), demands applying numerical optimization, since an analytic solution cannot be found.

12.4 Multilevel sample allocation for the scalarization estimator

Frequently, especially in the context of reliability optimization problems, both mean and standard deviation of the objective function are required. We have presented this measure as $\mathcal{R}_2^b$ in Section 9.1. It could be formulated as a multi-objective optimization problem, which leads to a Pareto front of the design. Another approach is to use a linear combination of mean and standard deviation. We denote this combination of statistics as *scalarization*:

$$\mathbb{S}[b(\mathbf{x}, \boldsymbol{\theta}] = \mathbb{E}[b(\mathbf{x}, \boldsymbol{\theta})] + \alpha\sigma[b(\mathbf{x}, \boldsymbol{\theta})], \tag{12.24}$$

where the weight α is introduced to control the variability of the solution.

In this context, formulating the MLMC estimator is straightforward, as it merely requires the summation of the estimators for the mean and standard deviation, as defined in the prior sections:

$$\widehat{\zeta}_{\mathrm{ML,biased}} = \widehat{\mu}_{1,\mathrm{ML}} + \alpha \widehat{\sigma}_{\mathrm{ML,biased}}. \tag{12.25}$$

The resource allocation—along with the yet-to-be-defined variance of the estimator—remains consistent with the previous scenarios, and we express it as:

$$\begin{aligned} \overset{*}{\mathbf{N}}^{\mathbb{S}} &= \arg\min_{\mathbf{N}^{\mathbb{S}}} C_T^{\mathbb{S}}, \\ \text{s.t. } \mathbb{V}\left[\widehat{\zeta}_{\mathrm{ML,biased}}\right] &= \epsilon_{\mathbb{S}}^2. \end{aligned} \tag{12.26}$$

where we now *target the scalarization*, minimizing the total cost $C_T^{\mathbb{S}} := \sum_{\ell=1}^{L} C_\ell N_\ell^{\mathbb{S}}$. As before, we only adapt the variance term, $\mathbb{V}\left[\widehat{\zeta}_{\mathrm{ML,biased}}\right]$, in the constraint and its target, $\epsilon_{\mathbb{S}}^2$, of Eq. (12.26) for the new scalarization term.

A major challenge is to obtain a traceable expression for the estimator variance. We expand it to

$$\mathbb{V}\left[\widehat{\zeta}_{\mathrm{ML,biased}}\right] = \mathbb{V}\left[\widehat{\mu}_{1,\mathrm{ML}}\right] + \alpha^2 \mathbb{V}\left[\widehat{\sigma}_{\mathrm{ML,biased}}\right] + 2\alpha \mathbb{C}\mathrm{ov}\left[\widehat{\mu}_{1,\mathrm{ML}}, \widehat{\sigma}_{\mathrm{ML,biased}}\right], \tag{12.27}$$

where we now have a correlation between estimators $\widehat{\mu}_{1,\mathrm{ML}}$ and $\widehat{\sigma}_{\mathrm{ML,biased}}$, which results in the covariance term between estimators, $\mathbb{C}\mathrm{ov}\left[\widehat{\mu}_{1,\mathrm{ML}}, \widehat{\sigma}_{\mathrm{ML,biased}}\right]$.

For $\mathbb{V}\left[\widehat{\mu}_{1,\mathrm{ML}}\right]$ and $\mathbb{V}\left[\widehat{\sigma}_{\mathrm{ML,biased}}\right]$, we employ the previous results from Eq. (6.24) and from Eq. (12.23), respectively. The covariance term,

$$\mathbb{C}\mathrm{ov}\left[\widehat{\mu}_{1,\mathrm{ML}}, \widehat{\sigma}_{\mathrm{ML,biased}}\right] = \sum_{\ell=1}^{L} \mathbb{C}\mathrm{ov}\left[\widehat{\mu}_{1,\ell} - \widehat{\mu}_{1,\ell-1}, \widehat{\sigma}_{\ell,\mathrm{biased}} - \widehat{\sigma}_{\mathrm{biased},\ell-1}\right], \tag{12.28}$$

however, requires additional derivations and approximations.

We can present the terms level-by-level as

$$\begin{aligned} &\mathbb{C}\mathrm{ov}\left[\widehat{\mu}_{1,\ell} - \widehat{\mu}_{1,\ell-1}, \widehat{\sigma}_{\ell,\mathrm{biased}} - \widehat{\sigma}_{\mathrm{biased},\ell-1}\right] = \\ &\mathbb{C}\mathrm{ov}\left[\widehat{\mu}_{1,\ell}, \widehat{\sigma}_{\ell,\mathrm{biased}}\right] + \mathbb{C}\mathrm{ov}\left[\widehat{\mu}_{1,\ell-1}, \widehat{\sigma}_{\mathrm{biased},\ell-1}\right] \\ &- \mathbb{C}\mathrm{ov}\left[\widehat{\mu}_{1,\ell}, \widehat{\sigma}_{\mathrm{biased},\ell-1}\right] - \mathbb{C}\mathrm{ov}\left[\widehat{\mu}_{1,\ell-1}, \widehat{\sigma}_{\ell,\mathrm{biased}}\right]. \end{aligned} \tag{12.29}$$

This term is challenging to assess because of the square root in the standard deviation estimator. We propose three distinctive approximations for this term, where each has its own challenges and benefits:

- An upper bound based on correlation, named covariance *Pearson* upper bound.
- An approximation based on the correlation between mean and variance, named covariance approximation with *Correlation Lift*.
- An approximation based on bootstrapping, named covariance approximation with the *Bootstrap* method.

We will employ the terms, marked here in italic, as a convenient shorthand for these different approximation strategies in the forthcoming results section. First, however, we will derive the different estimators for Eq. (12.28) or Eq. (12.29).

12.4.1 Covariance Pearson upper bound

A straightforward approach to approximate the expression from Eq. (12.28) is to leverage its relation with the Pearson correlation coefficient and to employ its upper bound. The Pearson correlation coefficient is defined in the multilevel setting as

$$\rho[\widehat{\mu}_{1,\mathrm{ML}}, \widehat{\sigma}_{\mathrm{ML,biased}}] = \frac{\mathbb{C}\mathrm{ov}\left[\widehat{\mu}_{1,\mathrm{ML}}, \widehat{\sigma}_{\mathrm{ML,biased}}\right]}{\sqrt{\mathbb{V}[\widehat{\mu}_{1,\mathrm{ML}}]\mathbb{V}[\widehat{\sigma}_{\mathrm{ML,biased}}]}}, \tag{12.30}$$

where we know that the coefficient is bounded by $-1 \leq \rho[\widehat{\mu}_{1,\mathrm{ML}}, \widehat{\sigma}_{\mathrm{ML,biased}}] \leq 1$ (see [317]).

The lower and upper bound on $\rho[\widehat{\mu}_{1,\mathrm{ML}}, \widehat{\sigma}_{\mathrm{ML,biased}}]$ can be utilized to establish an upper bound for (12.27) as

$$\mathbb{V}\left[\widehat{\zeta}_{\mathrm{ML,biased}}\right] \leq \mathbb{V}\left[\widehat{\mu}_{1,\mathrm{ML}}\right] + \alpha^2 \mathbb{V}\left[\widehat{\sigma}_{\mathrm{ML,biased}}\right] + 2|\alpha|\sqrt{\mathbb{V}[\widehat{\mu}_{1,\mathrm{ML}}]\mathbb{V}[\widehat{\sigma}_{\mathrm{ML,biased}}]}. \tag{12.31}$$

This has the advantage of a straightforward derivation and implementation with low additional computational cost. However, applying the Pearson correlation yields a considerably conservative estimate since we assume $\rho[\widehat{\mu}_{1,\mathrm{ML}}, \widehat{\sigma}_{\mathrm{ML,biased}}] = 1$, disregarding the fact that the covariance term can even be negative. Hence, we will consider two other, more sophisticated, approaches.

12.4.2 Covariance approximation with Bootstrap

For the second method, rather than providing an upper bound, we utilize bootstrapping to directly approximate the covariance instead of an upper bound. The concept of bootstrapping can be summarized as follows: the approach repeatedly draws samples with replacement from the existing dataset to compute the estimators in a replicable manner. This enables calculating estimates for various quantities such as the standard error or bias of an estimator. In our specific case, we employ bootstrapping to estimate the covariance between the terms in Eq. (12.28), as demonstrated in literature, see [106, 107].

Using the set of samples for each level $\ell = 1, ..., L$, we perform bootstrapping by drawing S new *bootstrapped* sets $\{Q_s^{(i)}\}_{i=1}^{N_\ell}$ with replacement from $\{Q^{(i)}\}_{i=1}^{N_\ell}$, where $s = 1, ..., S$. We can compute S estimators for the Bootstrap mean $\widehat{\mu}_{1,\ell}^{(s)} = \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_s^{(i)}$ and the Bootstrap standard deviation $\widehat{\sigma}_{\ell,\mathrm{biased}}^{(s)} = \sqrt{\frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell}(Q(i)_s - \widehat{\mu}_{1,\ell}^{(s)})^2}$ from these bootstrapped sets. Finally, we estimate the covariance as follows:

$$\mathbb{C}\mathrm{ov}[\widehat{\mu}_{1,\ell}, \widehat{\sigma}_{\ell,\mathrm{biased}}] \approx \frac{1}{S-1}\sum_{s=1}^{S}\left(\widehat{\mu}_{1,\ell}^{(s)} - \frac{1}{s}\sum_{i=1}^{S}\widehat{\mu}_{1,\ell}^{(s)}\right)\left(\widehat{\sigma}_{\ell,\mathrm{biased}}^{(s)} - \frac{1}{S}\sum_{i=1}^{S}\widehat{\sigma}_{\ell,\mathrm{biased}}^{(s)}\right). \tag{12.32}$$

On the one hand, the Bootstrap approximation is more demanding in the implementation compared to the Pearson correlation bound of Section 12.4.1. On the other hand, however, it offers the advantage of an approximation instead of an upper bound, which results in a better estimate for the covariance if the two estimators are not highly correlated. A disadvantage of the Bootstrap approach is the additional computational cost from resampling and estimating the Bootstrap statistics. We stress here, though, that resampling does not require a re-evaluation of the black box, Q, but a computation of the estimators in Eq. (12.32) with already evaluated samples. Thus, these cost can be neglected if the evaluation of the model itself is the main bottleneck and computationally very expensive.

12.4.3 Covariance approximation with Correlation Lift

Finally, as third approach, we demonstrate another approximation for the covariance term. This is based on a relationship formulated for the covariance of the mean and variance estimator, as referenced in [101, 248]:

Lemma 17. *The covariance of the unbiased sample estimators for the mean* $\widehat{\mu}_{1,\ell} = \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_\ell^{(i)}$ *and variance* $\widehat{\mu}_{2,\ell} = \frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell}(Q_\ell^{(i)} - \widehat{\mu}_{1,\ell})^2$ *is given as*

$$\mathbb{C}ov[\widehat{\mu}_{1,\ell}, \widehat{\mu}_{2,\ell}] = \frac{\mu_{3,\ell}}{N_\ell}. \tag{12.33}$$

where $\mu_{3,\ell} = \mathbb{E}[(Q_\ell - \mu_{1,\ell})^3]$ *is the third central moment.*

See C.11 for the proof.

In Eq. (12.29), it is evident that there are covariance terms involving estimators at different levels. Specifically, we encounter the terms $\mathbb{C}\text{ov}[\widehat{\mu}_{1,\ell}, \widehat{\mu}_{2,\ell-1}]$ and $\mathbb{C}\text{ov}[\widehat{\mu}_{1,\ell-1}, \widehat{\mu}_{2,\ell}]$, which do not adhere to the previous result from Lemma 17. We proceed to establish analogous relationships for these terms, where the distinction lies in a one-level difference between the estimators, while retaining a dependence on the samples.

Lemma 18. *The covariance of the unbiased sample estimators for the mean* $\widehat{\mu}_{1,\ell} = \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_\ell^{(i)}$ *and variance* $\widehat{\mu}_{2,\ell-1} = \frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell}(Q_{\ell-1}^{(i)} - \widehat{\mu}_{1,\ell-1})^2$ *is given as*

$$\begin{aligned}\mathbb{C}ov[\widehat{\mu}_{1,\ell}, \widehat{\mu}_{2,\ell-1}] = \frac{1}{N_\ell}\Big[&\mathbb{E}[Q_\ell(Q_{\ell-1})^2] - \mathbb{E}[Q_\ell]\mathbb{E}[(Q_{\ell-1})^2] \\ &- 2\mathbb{E}[Q_{\ell-1}]\mathbb{E}[Q_\ell Q_{\ell-1}] + 2\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}]^2\Big].\end{aligned} \tag{12.34}$$

See C.12 for the proof.

Lemma 19. *The covariance of the unbiased sample estimators for mean* $\widehat{\mu}_{1,\ell-1} = \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_{\ell-1}^{(i)}$ *and variance* $\widehat{\mu}_{2,\ell} = \frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell}(Q_\ell^{(i)} - \widehat{\mu}_{1,\ell})^2$ *is given as*

$$\begin{aligned}\mathbb{C}ov[\widehat{\mu}_{1,\ell-1}, \widehat{\mu}_{2,\ell}] = \frac{1}{N_\ell}\Big[&\mathbb{E}[Q_{\ell-1}(Q_\ell)^2] - \mathbb{E}[Q_{\ell-1}]\mathbb{E}[(Q_\ell)^2] \\ &- 2\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}Q_\ell] + 2\mathbb{E}[Q_{\ell-1}]\mathbb{E}[Q_\ell]^2\Big].\end{aligned} \tag{12.35}$$

The proof is the same as for Lemma 12.34 by interchanging ℓ and $\ell - 1$.

Lastly, in order to estimate the covariance term, we assume that the Pearson correlation coefficient between $\widehat{\mu}_1$ and $\widehat{\mu}_2$ exhibits similar behavior as the Pearson correlation coefficient between $\widehat{\mu}_1$ and $\widehat{\sigma}_{\text{biased}}$. Based on this assumption, we can employ the following relationship:

$$\begin{aligned}
&\rho[\widehat{\mu}_{1,\ell_i}, \widehat{\sigma}_{\ell_j,\text{biased}}] \approx \rho[\widehat{\mu}_{1,\ell_i}, \widehat{\mu}_{2,\ell_j}] \\
\Leftrightarrow\ &\frac{\mathbb{C}\text{ov}\left[\widehat{\mu}_{1,\ell_i}, \widehat{\sigma}_{\ell_j,\text{biased}}\right]}{\sqrt{\mathbb{V}[\widehat{\mu}_{1,\ell_i}]\mathbb{V}[\widehat{\sigma}_{\ell_j,\text{biased}}]}} \approx \frac{\mathbb{C}\text{ov}\left[\widehat{\mu}_{1,\ell_i}, \widehat{\mu}_{2,\ell_j}\right]}{\sqrt{\mathbb{V}[\widehat{\mu}_{1,\ell_i}]\mathbb{V}[\widehat{\mu}_{2,\ell_j}]}} \\
\Leftrightarrow\ &\mathbb{C}\text{ov}\left[\widehat{\mu}_{1,\ell_i}, \widehat{\sigma}_{\ell_j,\text{biased}}\right] \approx \mathbb{C}\text{ov}\left[\widehat{\mu}_{1,\ell_i}, \widehat{\mu}_{2,\ell_j}\right] \sqrt{\frac{\mathbb{V}[\widehat{\mu}_{2,\ell_j}]}{\mathbb{V}[\widehat{\sigma}_{\ell_j,\text{biased}}]}},
\end{aligned} \tag{12.36}$$

where $(\ell_i - \ell_j) \in \{-1, 0, 1\}$. As a result, we obtain an estimator for the different covariance terms in Eq. (12.29).

The Correlation Lift also offers an approximation, which shares the same advantages as the Bootstrap approximation compared to the Pearson upper bound. While implementing all the derived quantities is more demanding than bootstrapping, the computational cost of evaluating the quantities are much lower. Thus, for fast computational models, this approximation has advantages compared to Bootstrap.

Example 7 (Comparison of covariance approximations)**.** *We regard a two-level example where we compare the different covariance terms from Eq.* (12.29) *in Figure 12.1. The histogram shown in the four figures are plotted from repeatedly evaluating the different quantities. The reference solution in red is computed by repeatedly computing the covariance terms 2000 times from 1000 estimators for $\widehat{\mu}_{1,1}$, $\widehat{\mu}_{1,2}$, $\widehat{\sigma}_{1,biased}$ and $\widehat{\sigma}_{2,biased}$ each. Each of these estimator is computed with 100 samples. We compare against the Pearson upper bound as presented in Section 12.4.1 in orange, the Bootstrap approximation from Section 12.4.2, where we use $B = 1000$ bootstrap samples in green and the Correlation Lift approach from Section 12.4.3 in blue.*

We observe that the Pearson upper bound is indeed an upper bound and quite conservative, assigning a large value to the covariance terms. We see a good but not perfect match for the Bootstrap approach, where we also have to consider the computational cost of repeatedly computing the 1000 Bootstrap samples. Finally, we see a good match in the Correlation Lift, at least in the mean value of the histogram, while we see a larger variance compared to the reference solution. While the differences to the reference solution might seem quite large, especially for the Pearson upper bound, we point out that all the terms are combined with additions and subtractions. Hence, also numerical cancellations of very small and similar values have to be considered here.

This final example closes the section on the new estimators for the variance, standard deviation and scalarization. We present the importance of the chosen approximation for the covariance term of Eq. (12.27) in the results. The performance of the method relies heavily on this choice and we will compare all three different approximations in the results section. Next, we present how we can compute the resource allocation for these new estimators and talk about algorithmic details afterwards.

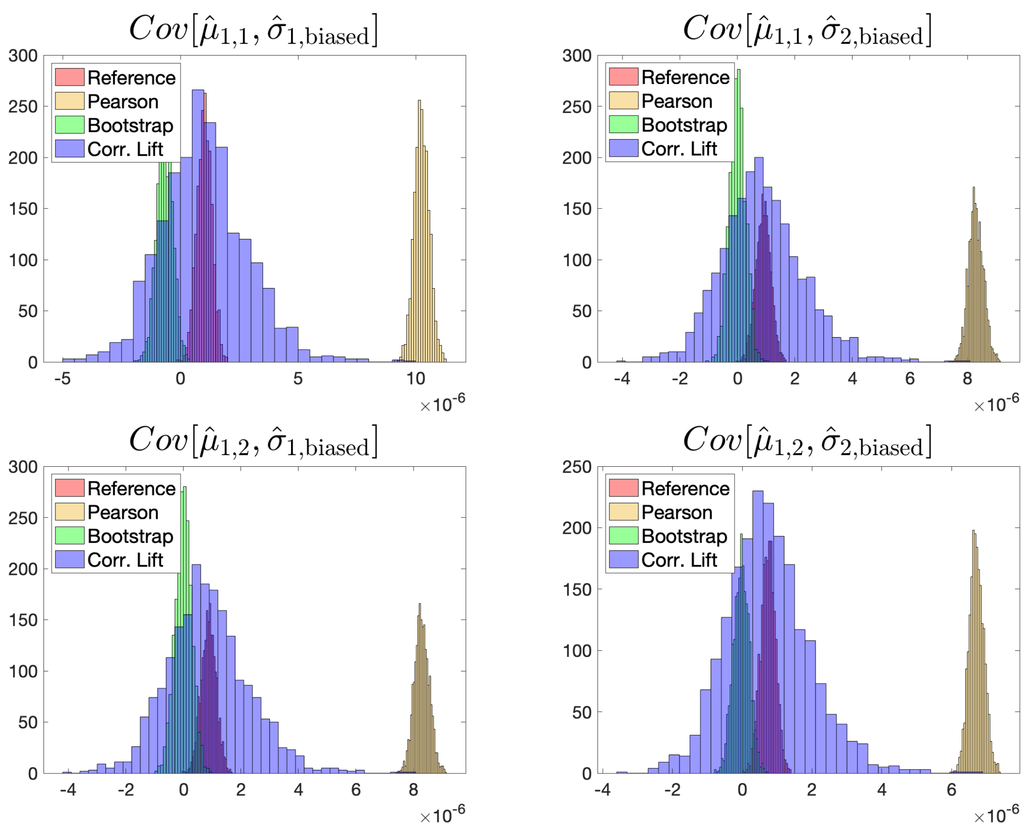


Figure 12.1: Comparison of the different covariance approximations for the four terms for a two level example in Eq. (12.29) to a numerically computed reference solution.

12.5 Analytic approximation

Although we can analytically solve the resource allocation problem for the mean estimator provided in Equation (6.26) using the approach described in Section 6.4, closed-form solutions are not available for the higher-order terms discussed in this work. Therefore, we resort to numerical optimization to obtain approximate solutions for the optimization problem. In addition, we incorporate an approach introduced in [189], where Krumscheid et al. propose an analytical approximation for the resource allocation problem involving higher-order central moments.

In their work, the authors make the key assumption that the variance of any higher-order sampling estimators, denoted as $\mathbb{V}[\widehat{\mu}_i]$ with $i \geq 2$, decreases at a rate of $\mathcal{O}(\frac{1}{N})$ with the number of samples denoted as N. Building upon this assumption, the authors introduce the variance estimator $\mathbb{V}[\widehat{\mu}_i] = \frac{\mathcal{V}[\widehat{\mu}_i]}{N}$, where the higher-order terms of N are now incorporated into $\mathcal{V}[\widehat{\mu}_i]$, while the explicit dependence on N remains in the term $\frac{1}{N}$. Consequently, $\mathbb{V}[\widehat{\mu}_i]$ exhibits the same structure as $\mathbb{V}[\widehat{\mu}_1] = \frac{\mathbb{V}[Q]}{N}$.

The authors extend this approach to the multilevel case: in a general setting for a multilevel central-moment estimator of higher order, $\widehat{\mu}_{i,\text{ML}}, i \geq 2$, its variance can be written as

$$\mathbb{V}[\widehat{\mu}_{i,\mathrm{ML}}] = \sum_{\ell=1}^{L} \frac{\mathcal{V}[\widehat{\mu}_{i,\ell}]}{N_\ell}, i \geq 2, \tag{12.37}$$

where the authors introduce the short-hand notation $\mathcal{V}[\widehat{\mu}_{i,\ell}] = \mathbb{V}[\widehat{\mu}_{i,\ell}]N_\ell$. We observe that we indeed see the same form as in the case for the mean, with a variance term divided by the number of samples.

Since we see the same structure, we apply the same approach as [136] to solve the problem analytically and find the Lagrange constant as

$$\lambda = \epsilon_{\mathbb{X}}^{-2} \sum_{\ell=1}^{L} \sqrt{\mathcal{V}[\widehat{\mu}_{i,\ell}]C_\ell}, \tag{12.38}$$

for different $\epsilon_{\mathbb{X}}, \mathbb{X} \in \{\mathbb{E}, \mathbb{V}, \sigma, \mathbb{S}\}$. Finally, we can derive the resource allocation as

$$N_\ell^{\mathbb{E}} = \left\lceil \lambda \sqrt{\frac{\mathcal{V}[\widehat{\mu}_{i,\ell}]}{C_\ell}} \right\rceil. \tag{12.39}$$

While the authors of [189] focus on higher-order central moments, we extend the aforementioned idea to approximate the variance of the standard deviation $\widehat{\sigma}_{\mathrm{ML,biased}}$ and the scalarization $\widehat{\zeta}_{\mathrm{ML,biased}}$. Whereas their work uses on h-statistics to derive general expressions for the variance of higher-order central moments, our contribution lies in direct formulations for the variances of the standard deviation and scalarization. Moreover, we utilize this approach in conjunction with numerical optimization for resource allocation computation. This approach allows us to solve the problem analytically by disregarding higher-order terms of N_ℓ, while the numerical optimization tackles the problem numerically, considering those terms. Hence, there is a trade-off between the approximation of the resource allocation and the approximation due to the numerical optimization. Both approaches also use, of course, only estimators instead of exact values for all the statistics involved.

This approach also offers flexibility from an algorithmic perspective. We have the option to directly employ the analytic approximation or use it as an initial guess for the numerical optimization. We compare and contrast these two approaches and different algorithmic choices in the result chapter of this part.

12.6 Algorithmic details

When computing the resource allocation problem, e.g., for the standard deviation in Eq. (12.22) or the scalarization in Eq. (12.26), we need to know all quantities required in the computations. Naturally, we do not know them in advance or can compute them with a high number of samples (which could be used to solve the problem itself). Hence, we resort to solving the resource allocation problem iteratively. We present the algorithm in the following:

The algorithm starts with a set of pilot samples, $\{\mathbf{N}^{\mathbb{X},(0)}\}_{\ell=1}^{L}$, which we evaluate to compute the required estimators for the SoI $\mathbb{X} \in \{\mathbb{E}, \mathbb{V}, \sigma, \mathbb{S}\}$. Using these estimators, we solve the resource allocation problem, Eq. (12.1), targeting $\mathbb{X}$ and compute a first estimate for the number of samples required on each level. We evaluate this suggested set of samples, update our estimators and again solve the resource allocation problem. We iteratively proceed until either the suggested resource allocation is smaller than the number of samples already evaluated, i.e. the increment is negative (or zero); or, we stop, when we reach a maximum number of iterations. We are going to compare the choice of iterations in the result section.

Additionally, for the higher order moments, we use an underrelaxation, since we have observed overshooting in the resource allocation for small numbers of pilot samples. Underrelaxation means that we only increase the current resource allocation by a factor $\xi \in (0, 1]$ of the suggested resource allocation. In the current implementation, ξ is linearly increasing with the number of iterations in the algorithm to account for better estimation when the number of pilot samples rises. After five iterations, we set $\xi = 1$.

Finally, in Section 12.5, we have noted that we can use the analytic approximation or the numerical optimization approach to solve the resource allocation problem for higher-order statistics. In the implementation, we have the option to use the analytic approximation only. Alternatively, we combine the two approaches, by using the analytic approximation as an initial guess for the numerical optimization to improve the performance of the optimizer. The numerical optimizer also switches to a logarithmic scaling if a solution cannot be found in a first iteration. We summarize the algorithm in Alg. 2.

Algorithm 2 Resource allocation algorithm targeting statistic $\mathbb{X}$.

```
1: Input: $\mathbb{X} \in \{\mathbb{E}, \mathbb{V}, \sigma, \mathbb{S}\}$, $\epsilon^2_{\mathbb{X}}$, $\mathbf{N}^{\mathbb{X},(0)} = [N_1^{\mathbb{X},(0)}, ..., N_L^{\mathbb{X},(0)}]$, $I_{\max}$, $j = 0$
2: $\triangle\mathbf{N}^{\mathbb{X},(0)} = \mathbf{N}^{\mathbb{X},(0)}$
3: while $j < I_{\max}$ or $\sum_{\ell=1}^{L} \triangle N_\ell^{\mathbb{X},(j)} > 0$ do
4:     Sample and evaluate black box $\{Q_\ell^{(i)}, Q_{\ell-1}^{(i)}\}_{i=1}^{\triangle N_\ell^{\mathbb{X},(j)}}$ for $\ell = 1, ..., L$
5:     Estimate necessary statistics and compute $\mathbb{V}[\hat{X}_{\mathrm{ML}}]$
6:     Solve Eq. (12.1) for $\mathbf{N}^{\mathbb{X},(j+1)}$ with $\epsilon^2_{\mathbb{X}}$ using analytic approximation
7:     if Use numerical optimization then
8:         Solve Eq. (12.1) for $\mathbf{N}^{\mathbb{X},(j+1)}$ with $\epsilon^2_{\mathbb{X}}$ using numerical optimization
9:     end if
10:    Compute underrelaxation factor: $\xi = (\mathbb{X} == \mathbb{E})?\xi = 1 : \xi = \min\left(\frac{j+1}{5}, 1\right)$
11:    Compute sample difference: $\triangle\mathbf{N}^{\mathbb{X},(j+1)} = \xi(\mathbf{N}^{\mathbb{X},(j+1)} - \mathbf{N}^{\mathbb{X},(j)})$
12:    Update iteration: $j = j + 1$
13: end while
14: Compute final statistics with resource allocation $\overset{*}{\mathbf{N}}{}^{\mathbb{X}} = \mathbf{N}^{\mathbb{X},(j)}$
```

We integrated the new MLMC algorithms and estimators we have developed in this work directly into Dakota. The presented methods can be set in the input file as `method`:

multilevel_sampling. With the setting pilot_samples, we set the initial samples for each level, $\mathbf{N}^{\mathbb{X},(0)} = [N_1^{\mathbb{X},(0)}, ..., N_L^{\mathbb{X},(0)}]$, and max_iterations sets the maximum iterations $I_{\max}$. Using convergence_tolerance, we specify the target accuracy of our estimator $\epsilon_{\mathbb{X}}^2$, whereas with convergence_tolerance_type, we specify if we want to target this convergence tolerance absolute or in a relative sense. Relative means, that we want to reduce $\epsilon_{\mathbb{X}}^2$ relative to the initial evaluation using the pilot samples. With convergence_tolerance_target, we can choose if we want to reduce cost targeting certain variance_constraint as in Eq. 6.26, but we can also reduce the variance targeting a certain cost_constraint as mentioned in Rem. 1. We pick the resource allocation target using allocation_target, where we can choose mean, variance, standard_deviation and scalarization. We activate the numerical optimization combined with the analytic approximation by setting the option optimization. The setting qoi_aggregation decides how the resource allocation for different functions is combined, where either the max value or a average over each level is taken. The cost of the different levels are set in solution_level_control, which is a model option.

Example 8 (Dakota input file for four-level case)**.** *The Dakota input file is given in Lst. 12.1. We estimate the scalarization statistic, $\mathbb{X} = \mathbb{S}$, in the resource allocation, where we target a variance of $\epsilon_{\mathbb{S}}^2 = 1.6175e - 05$ solving the optimizaton problem of Eq. 12.26. This examples uses 10 iterations and $\mathbf{N}^{\mathbb{S},(0)} = [10000, 1000, 100, 50]$ pilot samples for the lowest to the highest level. We use numerical optimization combined with the analytic approximation as initial guess. The cost for each level are given as $\{C_i = 10^{-4+i}\}_{i=1}^4$ from coarse to fine.*

```
method,
    id_method = 'UQ'
    model_pointer = 'HIERARCH'
        multilevel_sampling
      pilot_samples = 10000 1000 100 50
      sample_type random
      final_moments standard
      max_iterations = 10
      convergence_tolerance 1.6175e-05
        convergence_tolerance_type absolute
        variance_constraint
      allocation_target scalarization
        optimization
      qoi_aggregation max
```

```
model,
    id_model = 'MLModel'
    variables_pointer = 'UQ_V'
    interface_pointer = 'UQ_I'
    responses_pointer = 'UQ_R'
    simulation
        solution_level_control = 'Af'
```

```
            solution_level_cost = 0.01  1.0  0.1  0.001
```

Listing 12.1: Extract of Dakota input file for MLMC sampling simulation for four-level case.

For the numerical optimization of the resource allocation, Dakota uses the optimization library `OPT++` [227] by default. It is a nonlinear optimization package developed in C++, which provides a nonlinear interior-point methods for the optimization. Alternatively, Dakota offers linking to NPSOL [138] if a licence of the optimization library is available. NPSOL uses a sequential quadratic programming method for solving the optimization problem. NPSOL showed slightly improved performance in our tests compared to OPT++, mainly in a lower number of optimization iterations and, thus, faster convergence. We also provide gradients for the optimization of all resource allocation problems developed in this work. In the next example, we illustrate the objective function and constraint of a resource allocation problem for the variance of a two-level case.

Example 9 (Resource allocation problem for two-level case)**.** *For illustration purposes, we plot the objective function plus equality constraint of the resource allocation problem for a two-level problem in Figs. 12.2 and 12.3. We target the variance of Eq.* (12.10) *in this example. We present the surface and contour plot. We compare OPT++ and NPSOL, given in the title. Red circles show the optimization path. Red crosses show initial design, red diamonds show final design. The initial point is set as* $[N_1^{\mathbb{V},(0)}, N_2^{\mathbb{V},(0)}] = [5, 5]$ *to start in a region of high gradients. The number in title corresponds to a specific random seed. The equality constrained for* $\epsilon_{\mathbb{V}}^2$ *is visualized in magenta.*

We see that both approaches find an optimum on the equality constraint. NPSOL shows better performance, needing less optimization steps. Furthermore, we observe that the optimum lies in a region of low gradients, which motivates the switch to a logarithmic scaling of the objective function. We show more illustrations for different random seeds in App. C.13.

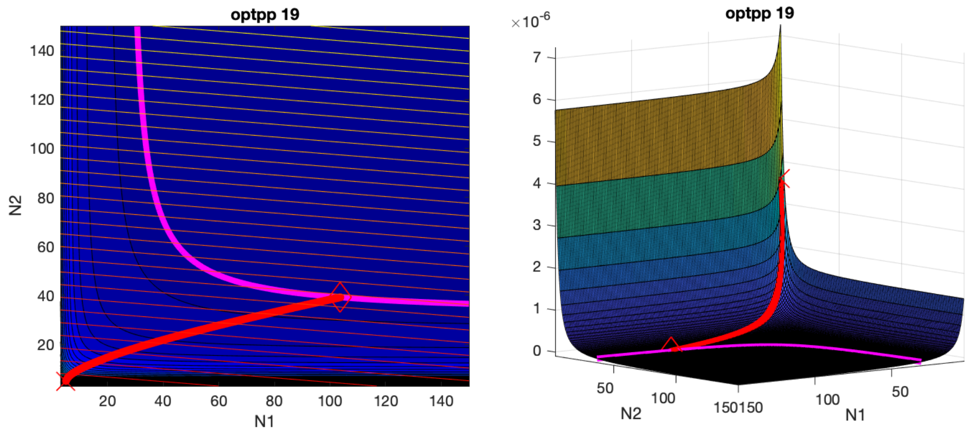


Figure 12.2: Surface and contour plot of the resource allocation objective function and variance constraint for OPTPP.

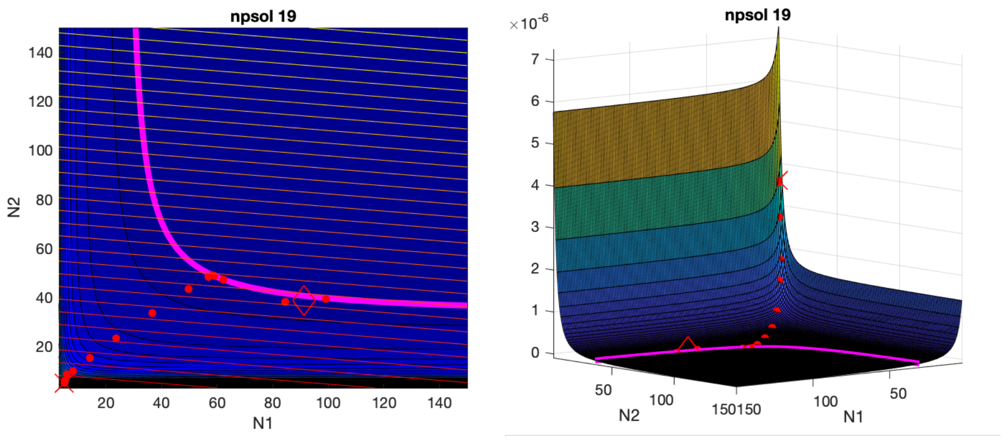


Figure 12.3: Surface and contour plot of the resource allocation objective function and variance constraint for NPSOL.

This closes the section on algorithmic details for the newly developed multilevel estimators in this work. The iterative resource allocation, the underrelaxation for higher-order moments and the choice of numerical optimizer all play a role in the performance of the method. These choices were motivated by benchmark results. We will present these benchmark results, verifying and validating the different estimators next.

13 Numerical results for a one-dimensional benchmark

In this chapter, we are going to evaluate the performance of the new MLMC estimators in the context of optimal resource allocation for different statistics. We will consider a one-dimensional problem, referred to as "Problem 18", which is discussed in detail in Section 13.1. We will extend the existing literature test cases to stochastic problems and incorporate multiple levels/approximations to define the MLMC estimators.

Using this example, we will assess the effectiveness of our MLMC estimators in evaluating various statistics, which corresponds to a forward UQ analysis with a fixed design. We will focus solely on the sampling aspect without including the outer loop of the OUU workflow. Moreover, we will numerically test how well the new estimators match their specific targets $\epsilon^2_{\mathbb{X}}$, where $\mathbb{X} \in \{\mathbb{E}, \mathbb{V}, \sigma, \mathbb{S}\}$. These targets will be computed from a MC solution with a predetermined computational cost as reference solution. This allows us to compare how well the newly developed MLMC statistics match their respective MC references.

Furthermore, we will compare the performance of our newly contributed MLMC estimators targeting different statistics to the standard MLMC estimator targeting the mean, each with its own sample allocation. Despite using different targets for the resource allocation, we aim to approximate a particular SoI. This comparison is essential to demonstrate that if we fix the MC cost and determine the corresponding precision for different targets (e.g., mean or standard deviation), the same MLMC sample profile that ensures the required accuracy in one statistic (e.g. mean) may not achieve the same precision in another statistic (e.g. standard deviation). This illustrates the need to match the target statistics of the MLMC estimator in the allocation process to the SoI in order to obtain the best allocation and desired accuracy.

Moreover, we will compare algorithmic choices in our analysis: first, we will compare the numerical optimization of the resource allocation problem to the approach adapted from [189], which was presented in Section 12.5. Second, we will compare the use of iterations, as described in Section 12.6. Third, we will highlight the impact of the covariance term approximation, discussed in Sections 12.4.1-12.4.3, on the efficiency of MLMC in the scalarization case. Throughout the results, we are going to demonstrate the improvements that MLMC can provide compared to its MC counterpart.

13.1 Problem 18 definition

For our test case, we choose to adapt problem 18 from the website in [131], which provides a collection of benchmark problems for optimization. In this scenario, we consider the one-dimensional deterministic function $f_{\text{det}} : \mathbb{R} \to \mathbb{R}$:

$$f_{\text{det}}(x) = \begin{cases} (x-2)^2 & \text{if } x \leq 3, \\ 2\ln(x-2)+1 & \text{if } x > 3. \end{cases} \tag{13.1}$$

Instead of optimization, we re-use this example for sampling first.

To introduce stochasticity into the problem, we incorporate a random variable $\theta \sim \mathcal{U}(-0.5, 0.5)$ and vary the correlation parameters to generate four distinct levels of f for the multilevel case. Consequently, we obtain the following four levels:

$$\begin{aligned} f_4(x,\theta) &= f_{det}(x) + \theta^3, \\ f_3(x,\theta) &= f_{det}(x) + 1.1\theta^3, \\ f_2(x,\theta) &= f_{det}(x) + \left(\frac{1}{60}x + 1.2\right)\theta^3, \\ f_1(x,\theta) &= f_{det}(x) + \frac{3}{2}\theta^3. \end{aligned} \tag{13.2}$$

Here, we designate f_4 as the finest resolution, while $\{f_i\}_{i=1}^3$ represents coarser levels with a computational cost hierarchy of $C_1 < C_2 < C_3 < C_4$. We maintain a cost ratio of $\frac{C_i}{C_{i-1}} = 10$, with $C_4 = 1$. Since the stochastic term has an additive nature, we can easily compute reference solutions. For instance, we have $\mathbb{E}[f_4(x,\theta)] = f_{\text{det}}(x)$ and $\mathbb{V}[f_4(x,\theta)] = \mathbb{V}[\theta^3] = \frac{0.5^6}{7}$.

13.2 Experimental setting

We use the following problem setting to evaluate the performance of our contribution. First, we focus on the sampling problem of estimating different measures for the objective function f_4 at a specific location x. Second, we compare the performance of the standard single-level MC estimator with the new MLMC estimators presented in the previous sections. Third, we evaluate the quality of estimation for the mean ($\widehat{\mu}_{1,\text{ML}}$), variance ($\widehat{\mu}_{2,\text{ML}}$), standard deviation ($\widehat{\sigma}_{\text{ML,biased}}$), and the scalarization term ($\widehat{\zeta}_{\text{ML,biased}} = \widehat{\mu}_{1,\text{ML}} + 3\widehat{\sigma}_{\text{ML,biased}}$). Finally, we compare various algorithmic choices.

We assess the following algorithmic choices, where the notation for the legends in the upcoming figures is given in parentheses:

- The impact of iteratively computing the resource allocation for all estimators in a single iteration (*1 iter*) or 20 iterations (*20 iter*), as described in Section 12.6.
- The comparison between using numerical optimization (*Opt*) and the analytic approximation (*AA*) extended from [189] and presented in Section 12.5 to compute the resource allocation for the newly developed estimators.
- The comparison of the covariance approximation as described in Sections 12.4.1 (*Pearson*), 12.4.3 (*CorrLift*), and 12.4.2 (*Bootstrap*) for the scalarization term $\widehat{\zeta}_{\text{ML,biased}}$.

Without loss of generality, we fix the location $x = 1$ and repeatedly compute the estimators 1000 times with a random seed and, each time, compute the respective resource allocation. From those samples of estimators, we plot histograms to show their distribution; additionally, we compute the mean and variance of the distributions. We compare the different MLMC approaches on different targets among each other and also to a single level MC.

By fixing $\epsilon^2_{\mathbb{X}}, \mathbb{X} \in \{\mathbb{E}, \mathbb{V}, \sigma, \mathbb{S}\}$, to be equal to the respective variance of the MC reference solution (based on 1000 samples), we expect the MLMC estimators to match the performance of the single level MC at a reduced cost, since both target the same variance. This reference variance for 1000 samples is computed analytically for this test case, e.g. for the mean: $\epsilon^2_{\mathbb{E}} = \frac{\mathbb{V}[f_4(x,\xi)]}{1000} = \frac{0.5^6}{7000} \approx 2.2321\text{e-}6$, or numerically for higher-order statistics such as the scalarization.

The results are computed using Dakota, using the coupling as described in Section 12.6. The different problem definitions are defined in Dakota input files. We give an example of the input file for Dakota for the scalarization case in the appendix in Lst C.1.

13.3 Results

We will discuss the results for the different estimators that we developed in this work next. We will start with the mean, as proof of concept and to gain familiarity with the results. Afterwards, we will present results for the variance, standard deviation and scalarization.

For all results, we will present histogram plots to see a qualitative match in estimators and tables to show the quantitative error in approximation. For all of the following histograms, we compare the MC reference solution in red, with the respective MLMC estimator denoted in the legend. The estimated SoI is given in the figure title whereas the targeted resource allocation of the MLMC estimator is given in its legend entry. The algorithmic choices of a single iteration (*1 iter*) or 20 iterations (*20 iter*) and numerical optimization (*Opt*) or the analytic approximation (*AA*) are also given in the legend using the abbreviations in parentheses. The computational cost are computed relative to the MC reference solution, which has a cost of 1000 samples on ℓ_4 with $C_4 = 1$. The total cost for the MLMC estimators are computed as $C^{\mathbb{X}}_T = \sum_{\ell=1}^{L} \overset{*}{N}{}^{\mathbb{X}}_{\ell} C_\ell, \mathbb{X} \in \{\mathbb{E}, \mathbb{V}, \sigma, \mathbb{S}\}$.

13.3.1 Mean

In the first case, we estimate the expectation, $\widehat{\mu}_1$. We use $\epsilon^2_{\mathbb{E}} \approx 2.2321\text{e-}6$ as the target variance $\mathbb{V}[\widehat{\mu}_{1,\mathrm{ML}}]$ to compute the resource allocation. The resulting histogram, obtained from 1000 independently computed estimators, is presented in Fig. 13.1 on the left.

The distributions, depicted in red (reference MC solution) and blue (MLMC estimator) as described in [136], exhibit a clear match. We also explore the impact of the number of iterations on the estimation quality. By performing 20 iterations (20 iter) to determine the resource allocation, we observe a closer match with the reference solution compared to a single iteration (1 iter) of the algorithm. This improvement arises from the reduced variance in the estimator statistic introduced by the decreased dependence on the pilot

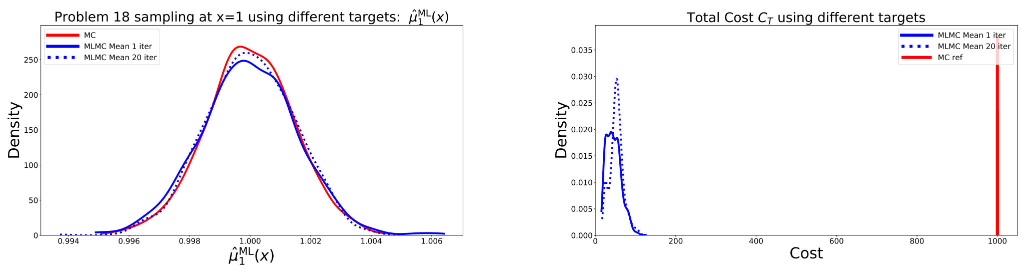


Figure 13.1: Mean. Left: Histogram over 1000 samples of $\widehat{\mu}_{1,\mathrm{ML}}$ for $x = 1$ using the different estimators described in Eq. (6.26) in blue compared to a reference MC estimator in red. The resource allocation problem is solved analytically. Right: Respective cost for the different estimators.

sampling compared to the single-iteration case. In the iterated approach, more samples are added iteratively, refining the statistics until convergence.

Furthermore, analyzing the computational cost on the right of Fig. 13.1, we see that the MLMC approach yields a significant reduction in cost compared to the single-level MC solution. When comparing the number of iterations, we also observe a narrower peak in the cost distribution for 20 iterations. This indicates a more robust computational cost when using the iterative approach as compared to a single iteration.

Table 13.1 compares the expectations and variances computed from the estimator histograms. We also know the exact target values for both the expectation of our estimators and their variances. When we quantitatively evaluate the performance, it is evident that the MLMC Mean (20 iter) provides the best approximation of the expected value, while also achieving a variance of the estimator that is closest to the target.

Method	Mean	Exact	Variance	Exact
MC	0.99999516		2.1271e-6	
MLMC Mean (1 iter)	0.99996403	1.0	2.5850e-6	2.2321e-6
MLMC Mean (20 iter)	0.99999945		2.1821e-6	

Table 13.1: Expectations and variances from the histograms of the different approaches in Fig. 13.1. The column labeled *Exact* shows the target value for expectation and variance.

13.3.2 Variance

In the second case, we focus on estimating the variance $\widehat{\mu}_2$, where we compute the resource allocation using $\epsilon_{\mathbb{V}}^2 \approx 1.3823\text{e-}8$ as the target for its variance $\mathbb{V}[\widehat{\mu}_{2,\mathrm{ML}}]$. This target is analytically computed for a MC reference solution using 1000 samples. We compare the result not only to a MC reference solution but also to an MLMC estimator targeting the mean. In this case, similar to the previous case, the MLMC estimator targeting the mean uses $\epsilon_{\mathbb{E}}^2 \approx 2.2321\text{e-}6$ as the target for its variance $\mathbb{V}[\widehat{\mu}_{1,\mathrm{ML}}]$ as defined in its resource allocation problem from Eq. (6.26). We first compare the performance to the standard MC estimator and to the MLMC mean estimator before contrasting different algorithmic choices for the MLMC variance estimator.

The resulting histogram, obtained from 1000 independently computed estimators, is shown in Fig. 13.2. Here, we only compare it to the MLMC mean estimator on the left. This is the first case where we observe the importance of allocating resources based on the SoI. While seeing a good match between MLMC estimator targeting the variance with the MC reference, we notice that the MLMC estimator targeting the mean is under-resolving the estimator, resulting in a much wider and shallower peak. This wider peak indicates that the resource allocation is underestimated. Consequently, the standard MLMC approach has a significantly lower computational cost (note the logarithmic scale on the x-axis). This clearly demonstrates the advantage of synchronizing the allocation target with the SoI. Allocating resources based solely on the mean cannot be expected to yield accurate MLMC estimators for other statistics. Furthermore, reducing computational cost compared to MC does not necessarily indicate that an estimator is advantageous.

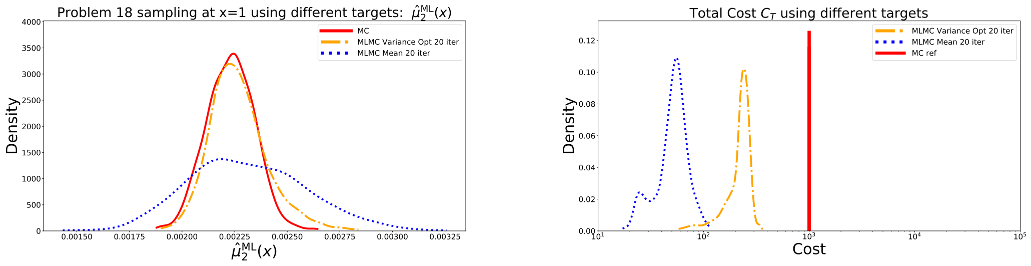


Figure 13.2: Variance. Left: Histogram over 1000 samples of $\widehat{\mu}_{2,\mathrm{ML}}$ for $x = 1$ comparing the new estimator described in Eq. (12.10) (orange, dashed dot) compared to a reference MC estimator (red, solid) and using the standard MLMC estimator targeting the mean (blue, dotted). Right: Respective cost for the different estimators.

Regarding the algorithmic implementations for the variance, we compare using the analytic approximation (AA), as described in Section 12.5, to using a combination of the analytic approximation with numerical optimization (Opt) in Fig. 13.3. Furthermore, for both options, we can use either a single iteration (1 iter) or 20 iterations (20 iter). Firstly, we observe the advantage of the iterative approach in reducing the variance in the cost distribution of the estimator in the plot on the right. Secondly, using numerical optimization in addition to the analytic approximation provides a slight improvement in the approximation quality for the estimator targeting $\epsilon_{\mathbb{V}}^2$, visible in the left plot.

This is also evident in the quantitative results presented in Table 13.2. Our MLMC estimator for the variance demonstrates a closer match to the exact solution, both in terms of its expectation and the variance of the estimator itself. When comparing the different approaches, we observe that MLMC Variance AA (1 iter) appears to perform best. However, we must also consider the computational cost. It is worth noting the high variance in the cost associated with this approach, which sometimes exceeds the cost of

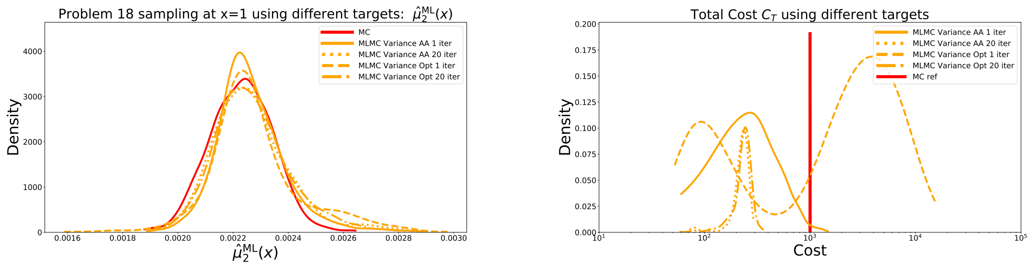


Figure 13.3: Variance. Left: Histogram over 1000 samples of $\widehat{\mu}_{2,\mathrm{ML}}$ for $x = 1$ comparing different algorithmic choices for computing the new estimators described in Eq. (12.10) in orange compared to a reference MC estimator in red. Right: Respective cost for the different estimators.

the standard MC method. Consequently, we continue to prefer the iterative approaches for their improved robustness in terms of computational cost.

Method	Mean	Exact	Variance	Exact
MC	2.2311e-3	2.2321e-3	1.3112e-8	1.3823e-8
MLMC Mean (20 iter)	2.3093e-3		7.6308e-8	
MLMC Variance AA (1 iter)	2.2656e-3		1.5904e-8	
MLMC Variance AA (20 iter)	2.2660e-3		2.0144e-8	
MLMC Variance Opt (1 iter)	2.2799e-3		2.8185e-8	
MLMC Variance Opt (20 iter)	2.2680e-3		1.8959e-8	

Table 13.2: Expectations and variances from the histograms of the different approaches in Fig. 13.2 and Fig 13.3. The column labeled *Exact* shows the target value for expectation and variance.

13.3.3 Standard deviation

While estimators for the variance have previously been presented in the work by [189] (using h-statistics), we now shift our focus to the standard deviation $\widehat{\sigma}_{\mathrm{biased}}$ in this third case. We utilize $\epsilon_\sigma^2 \approx 1.5493\text{e-}6$ as the target for the variance $\mathbb{V}[\widehat{\sigma}_{\mathrm{ML,biased}}]$ to compute the resource allocation. Since there is no exact analytical solution available (without resorting to approximations like the Delta method), we numerically compute this target by repeatedly estimating the standard deviation for 1000 samples and computing its variance over 1000000 repetitions. Similarly, when targeting the mean, we use $\epsilon_{\mathbb{E}}^2 \approx 2.2321\text{e-}6$, as mentioned in the results for the expected value. In this case, we adapt the analytic approximation introduced by [189] to these new estimators, as described in Section 12.5.

The resulting histograms, obtained from 1000 independently computed estimators, are depicted in Fig. 13.4 and Fig. 13.5 on the left. Akin to the variance case, we observe in Fig. 13.4 that the MLMC estimator targeting the mean is not well-suited for estimating the standard deviation. On the other hand, the MLMC estimator computed using Equation (12.22) demonstrates a good match with the single-level MC estimator.

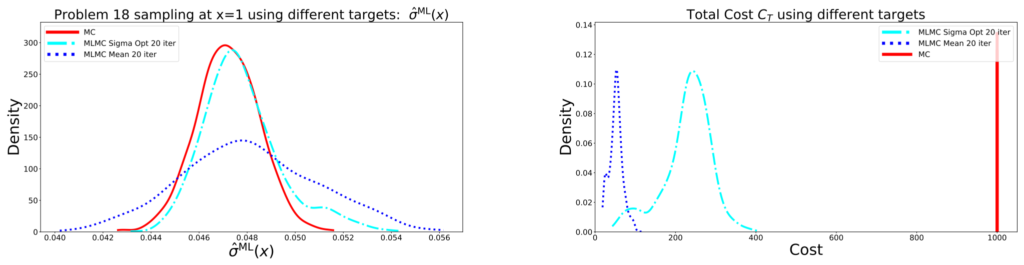


Figure 13.4: Standard deviation. Left: Histogram over 1000 samples of $\widehat{\sigma}_{\mathrm{ML,biased}}$ for $x = 1$ comparing the new estimator described in Eq. (12.22) (cyan, dashed dot) compared to a reference MC estimator (red, solid) and using the standard MLMC estimator targeting the mean (blue, dotted). Right: Respective cost for the different estimators.

In Fig. 13.5, we highlight the advantages of using an iterative approach combined with numerical optimization, particularly in terms of computational cost. When employing only a single iteration, the computational cost exhibits high variability and can even exceed the cost of MC. Conversely, the iterative approach results in a smaller variance in the cost. We also note that adapting the analytic approximation, as described in Section 12.5, combined with iterations yields satisfactory results in this case.

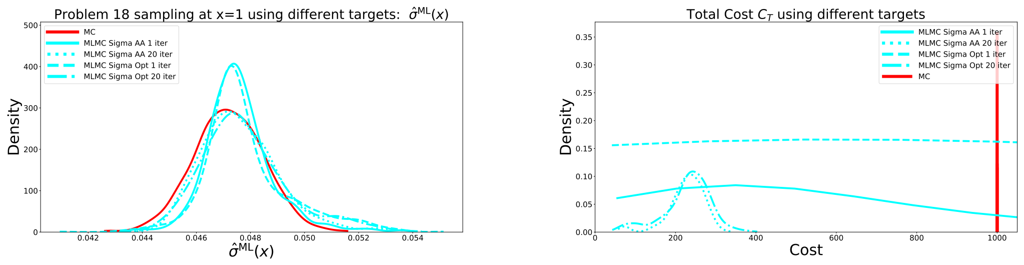


Figure 13.5: Standard deviation: Left: Histogram over 1000 samples of $\widehat{\sigma}_{\mathrm{ML,biased}}$ for $x = 1$ comparing different algorithmic choices for computing the new estimators described in Eq. (12.22) in cyan compared to a reference MC estimator in red. Right: Respective cost for the different estimators.

The results shown above are further supported by the quantitative analysis of the histogram expectations and variances (which approximate the variance of the estimator) presented in Table 13.3. We clearly observe superior performance in terms of the expectation and variance of the estimator for the newly developed MLMC estimator for the standard deviation. As a side note, the bias in the estimator $\widehat{\sigma}_{\mathrm{ML,biased}}$ is also apparent, resulting in a small offset compared to the reference solution.

Method	Mean	Exact	Variance	Exact
MC	4.7198e-2	4.7246e-2	1.6967e-6	1.5493e-6
MLMC Mean (20 iter)	4.8073e-2		7.8906e-6	
MLMC Sigma AA (1 iter)	4.7601e-2		1.6047e-6	
MLMC Sigma AA (20 iter)	4.7598e-2		2.0572e-6	
MLMC Sigma Opt (1 iter)	4.7848e-2		2.6471e-6	
MLMC Sigma Opt (20 iter)	4.7785e-2		2.6936e-6	

Table 13.3: Expectations and variances from the histograms of the different approaches in Fig. 13.4 and Fig 13.5. The column labeled *Exact* shows the target value for expectation and variance.

13.3.4 Scalarization

Lastly, we present the results for the new scalarization estimator, $\widehat{\zeta}_{\text{ML,biased}} = \widehat{\mu}_{1,\text{ML}} + \alpha\widehat{\sigma}_{\text{ML,biased}}$. For the computation of the resource allocation, we select $\alpha = 3$, which yields $\epsilon_{\mathbb{S}}^2 \approx$ 1.6175e-5 as the target for the variance $\mathbb{V}[\widehat{\zeta}_{\text{ML,biased}}]$. Similar to previous cases, this target is determined numerically by repeatedly estimating the estimator using 1000000 samples and computing its variance. We compare three different approaches for computing the covariance term, as described in Section 12.4. The comparison of the covariance approximation as described in Section 12.4.1.

Pearson correlation

In the first case, we utilize the Pearson correlation to compute the covariance term of $\mathbb{V}[\widehat{\zeta}_{\text{ML,biased}}]$, as described in Section 12.4.1. The results are depicted in Fig. 13.6. We observe the impact of employing the upper bound for estimating the variance: the estimators are over-resolved, leading to a smaller variance compared to the target. While we still achieve a good match with the reference solution, this conservative approximation results in an unnecessary computational cost. It is important to note once again that we observe the effect of the biased estimator arising from the estimation of the standard deviation.

Bootstrap approximation

In the second case, we employ the Bootstrap approximation to compute the covariance term of $\mathbb{V}[\widehat{\zeta}_{\text{ML,biased}}]$, as described in Section 12.4.2. Instead of a conservative upper bound, we now utilize an approximation approach. The improvement resulting from this choice is evident in Fig. 13.7: we observe a closer match between the histogram and the target function in the left figure. Additionally, the best results are achieved when using 20 iterations and numerical optimization.

When we examine the computational cost on the right, we notice that it is lower compared to Fig. 13.6, thanks to the approximation rather than the use of an upper bound. However, it is worth noting that the computational cost associated with computing

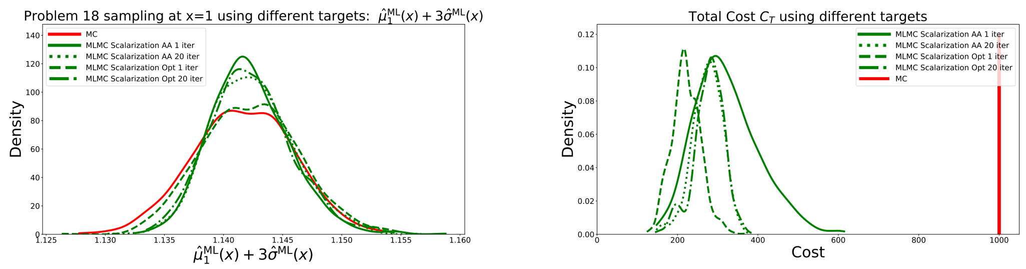


Figure 13.6: Scalarization (Pearson). Left: Histogram over 1000 samples of $\widehat{\zeta}_{\mathrm{ML,biased}}$ for $x = 1$ using the scalarization estimator described in Eq. (12.26) in green in combination with using the Pearson correlation property described in Section 12.4.1 to bound the covariance term in Eq. (12.27). We compare to a MC reference estimator in red. Right: Respective cost for the different estimators.

the Bootstrap estimator is not visualized here. This cost becomes especially significant when combined with numerical optimization. While it may still be relatively small when applied to expensive black-box functions, it becomes non-negligible compared to the evaluation of analytic functions.

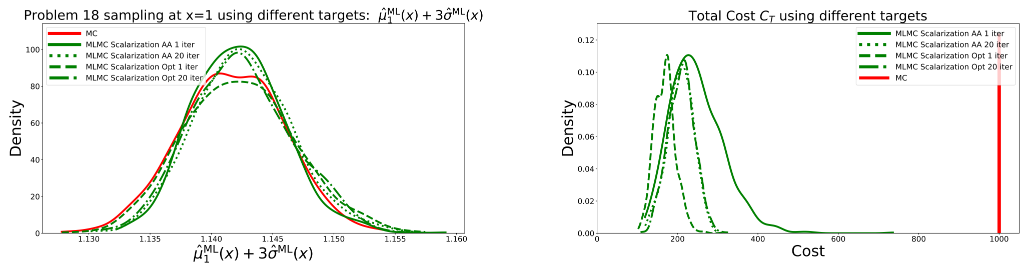


Figure 13.7: Scalarization (Bootstrap). Left: Histogram over 1000 samples of $\widehat{\zeta}_{\mathrm{ML,biased}}$ for $x = 1$ using the scalarization estimator described in Eq. (12.26) in green in combination with using the Bootstrap approximation described in Section 12.4.2 to approximate the covariance term in Eq. (12.27). We compare to a MC reference estimator in red. Right: Respective cost for the different estimators.

Correlation Lift

To address the concerns of having a conservative estimate for the Pearson bound or inflated computational cost for the Bootstrap approximation, we investigate the third case of the Correlation Lift. It which involves estimating the covariance term using the relationship between the covariance of the mean and variance, as described in Section 12.4.3. The results are presented in Fig. 13.8.

Once again, we observe a strong alignment between the histograms and the reference solution, while achieving similar computational cost compared to the Bootstrap approach in terms of the number of samples. However, the computational cost associated with evaluating the covariance term itself is significantly lower than that of repeatedly evaluating the Bootstrap term. Those cost do not show in the cost diagram.

Overall, it appears to be the most efficient strategy, as it offers the advantageous features of both approaches. It combines the low computational overhead of Pearson's correlation with the high approximation quality of the Bootstrap method.

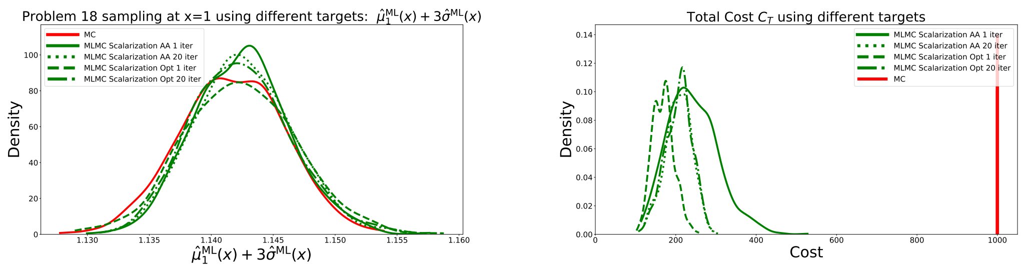


Figure 13.8: Scalarization (Correlation Lift): Left: Histogram over 1000 samples of $\widehat{\zeta}_{\mathrm{ML,biased}}$ for $x = 1$ using the scalarization estimator described in Eq. (12.26) in green in combination with using the Correlation Lift approximation described in Section 12.4.3 to bound the covariance term in Eq. (12.27). We compare to a MC reference estimator in red. Right: Respective cost for the different estimators.

Comparison with the mean

In the final analysis, we include the MLMC estimator targeting the mean in the resource allocation for the evaluation of the scalarization. The corresponding results are illustrated in Fig. 13.9. For clear comparison, we again use $\epsilon_{\mathbb{E}}^2 \approx 2.2321\text{e-}6$, as in the first case. In this scenario, we employ the Correlation Lift for approximating the scalarization and only present the results for 20 iterations combined with numerical optimization, for clarity.

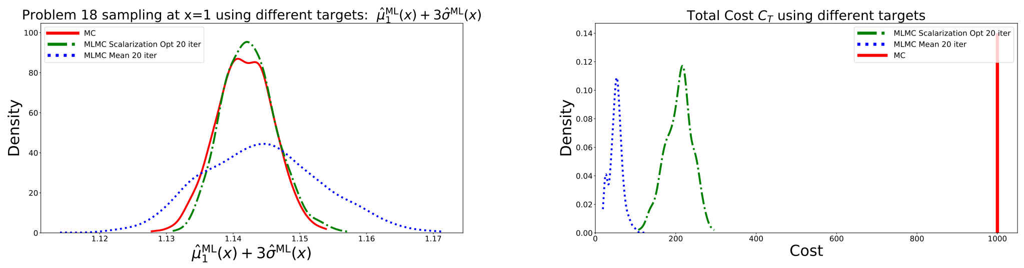


Figure 13.9: Scalarization (Correlation Lift including mean). Left: Histogram over 1000 samples of $\widehat{\zeta}_{\mathrm{ML,biased}}$ for $x = 1$ using the different estimators described in Eq. (6.26) in blue and Eq. (12.26) in green compared to a reference MC estimator in red. The covariance term of Eq. (12.27) is approximated using Correlation Lift approximation described in Section 12.4.3. We use numerical optimization and 20 iterations to find the resource allocation. Right: Respective cost for the different estimators.

A notable distinction becomes apparent when comparing the MLMC estimator targeting the mean with the other approaches. It significantly under-resolves the estimation, resulting in a much larger variance at a lower computational cost. It becomes evident that the allocation of resources in this case is insufficient to achieve the desired accuracy level compared to the MC reference.

Tables 13.4 and 13.5 provide quantitative comparisons for the scalarization case. These tables directly compare the three different approaches for approximating the covariance, as well as their algorithmic implementations. Firstly, we reiterate that all approaches yield improved results compared to using the standard MLMC estimator targeting the mean. This reinforces the qualitative findings presented in Fig. 13.9.

Furthermore, while the approximation qualities of the newly developed MLMC estimators targeting the scalarization are very similar, we do observe minor differences in the variance of the estimator. Similar to the previous results, employing a single iteration appears to yield a close match to the target variance when using the Pearson approximation. However, using 20 iterations tends to over-resolve the problem, resulting in a smaller variance but higher computational cost. For both the Bootstrap and Correlation Lift approaches, we find that using 20 iterations improves the variance approximation.

	Mean			
	Pearson	Bootstrap	Correlation Lift	Exact
MC	1.1417			1.1417
MLMC Mean (20 iter)	1.1443	1.1443	1.1443	
MLMC Scalarization AA (1 iter)	1.1422	1.1423	1.1425	
MLMC Scalarization AA (20 iter)	1.1424	1.1426	1.1426	
MLMC Scalarization Opt (1 iter)	1.1423	1.1423	1.1423	
MLMC Scalarization Opt (20 iter)	1.1423	1.1424	1.1424	

Table 13.4: Expectations from the histograms of the different approaches in Fig. 13.6, 13.7, 13.8 and Fig 13.9. The column labeled *Exact* shows the target value for the expectation.

	Variance			
	Pearson	Bootstrap	Correlation Lift	Exact
MC	1.7739e-5			1.6175e-5
MLMC Mean (20 iter)	1.1259e-5	7.7175e-5	7.7175e-5	
MLMC Scal. AA (1 iter)	1.6047e-5	1.3927e-5	1.4802e-5	
MLMC Scal. AA (20 iter)	1.1947e-5	1.5989e-5	1.5708e-5	
MLMC Scal. Opt (1 iter)	1.6534e-5	2.0454e-5	2.0591e-5	
MLMC Scal. Opt (20 iter)	1.1998e-5	1.6989e-5	1.6451e-5	

Table 13.5: Variances from the histograms of the different approaches in Fig. 13.6, 13.7, 13.8 and Fig 13.9. The column labeled *Exact* shows the target value for the variance.

Based on these results, the Correlation Lift approximation demonstrates the best performance and closely matches the target variance, while also considering computational cost. Moreover, there is a slight improvement when using numerical optimization compared to the analytic approximation.

Resource allocation profiles

We are going to conclude this section by examining the resource allocation profiles across levels: in Table 13.6, we present the average resource allocation, specifically the number of samples, for each level. The values are normalized with respect to the number of samples used at the finest level. The table displays the allocations for different statistics, including variance, standard deviation, and scalarization. We compare the standard MLMC approach, which targets the mean, with our presented MLMC approaches that target the respective statistics.

Statistic	Estimator Target	Level 1	Level 2	Level 3	Level 4
Variance	MLMC Mean	445.47	25.50	3.37	1
	MLMC Variance	386.51	42.82	3.32	1
Sigma	MLMC Mean	452.31	25.92	3.43	1
	MLMC Sigma	404.12	43.93	3.38	1
Scalarization	MLMC Mean	462.86	26.81	3.58	1
	MLMC Scalarization	359.52	32.42	3.27	1

Table 13.6: We present the averaged and normalized sample profiles for different statistics, comparing them to the standard MLMC estimator targeting the mean. In all cases, we have used 20 iterations and numerical optimization for the MLMC estimators targeting variance, standard deviation, and scalarization.

The key observation from this table is that we cannot simply scale the resource allocation of the standard MLMC targeting the mean when considering other statistics. Instead, we observe a redistribution of samples. Notably, there is an increase in samples allocated to the second level, while samples allocated to the first and third levels decrease. Therefore, it is crucial to adapt the MLMC target to the specific SoI. Simply scaling the standard MLMC estimator for the mean to vary its precision is insufficient and does not capture the optimal allocation for different statistics.

13.3.5 Conclusions

Based on our findings, it is evident that using the appropriate MLMC estimators targeting the SoI is crucial for achieving the expected accuracy at the lowest computational cost. We have demonstrated that relying solely on the MLMC estimator for the mean is insufficient to reach a specific target. Conversely, approaches that align the OUU goal with the allocation target are capable of achieving the desired accuracy.

Regarding algorithmic choices, we consistently obtained the best results when employing an iterative approach combined with numerical optimization to determine the resource allocation. This iterative process enhances the accuracy and robustness of the estimators, leading to improved performance.

Additionally, we have highlighted the significance of selecting an appropriate covariance approximation for the scalarization case. On the one hand, while Pearson correlation may appear convenient and straightforward, it tends to over-resolve the estimator; this results in unnecessary computational cost. On the other hand, bootstrapping yields good results but incurs additional computational cost due to resampling. As a balanced approach, we have shown that the correlation lift method presented in Section 12.4.3 strikes a favorable compromise in terms of performance and computational efficiency.

Part IV

The derivative-free stochastic nonlinear constrained optimization method SNOWPAC

...nothing at all takes place in the universe in which
some rule of maximum or minimum does not appear.
—*Leonhard Euler [115]*

14 Extending NOWPAC to stochastic problems

After having presented the deterministic derivative-free method NOWPAC [23] in Section 8.3 of Part II, we are going to introduce our second contribution of this work, SNOWPAC (stochastic NOWPAC), in the upcoming part. SNOWPAC is an extension of NOWPAC as a stochastic derivative-free optimization method for OUU. It is able to optimize constraint and computationally expensive black-box problems under uncertain conditions. Let us first recapitulate the optimization problems that we consider:

In NOWPAC, we intended to solve deterministic optimization problems of the form of Eq. (8.1), which we repeat here:

$$\begin{aligned} &\min f(\mathbf{x}) \\ \text{s.t.} \quad &c_i(\mathbf{x}) \leq 0, i = 1, ..., r. \end{aligned} \tag{14.1}$$

In the OUU framework, instead of solving a deterministic $f(\mathbf{x})$, we solve a robust formulation as we have discussed in Chap. 9. Here, we introduce the stochastic parameter $\boldsymbol{\theta}$ and solve Eq (9.1), which was given as:

$$\begin{aligned} &\min \mathcal{R}^f(\mathbf{x}, \boldsymbol{\theta}) \\ \text{s.t.} \quad &\mathcal{R}^{c_i}(\mathbf{x}, \boldsymbol{\theta}) \leq 0, i = 1, ..., r. \end{aligned} \tag{14.2}$$

In the next sections, we will answer the following questions:

- How do we compute the robustness measures $\mathcal{R}$ we discussed in Section 9.1?
- How do we adapt NOWPAC to be able to solve problems of the form of Eq. (14.2)?

We will first look at the statistical approximation of $\mathcal{R}$ in Section 14.1, where we employ sampling estimators. Afterwards, we will discuss which challenges ensue because of these approximations and which extension we add to NOWPAC because of that. These extensions include a noise adapted trust-region management in Section 14.2, Gaussian process supported noise correction in Section 14.3 and a feasibility restoration mode in Section 14.4. Afterwards, we will present the SNOWPAC algorithm itself in Chap. 15. We finish this part with presenting numerical results on the CUTEst benchmark set in Chap. 16. The following part is based on our work in [221].

14.1 Statistical estimation of robustness measures

First, we observe that the presented measures for robustness and risk from Section 9.1 can be written in terms of an expectation,

$$\mathcal{R}^b(\mathbf{x}) := \mathbb{E}_{\boldsymbol{\theta}}\left[B(\mathbf{x}, \boldsymbol{\theta})\right], \tag{14.3}$$

where the function B depends on the actual choice of measure. Throughout this work we assume that B has finite variance. We present the integrand of different measures in Table 14.1.

Measure	Eq.	integrant $B(\mathbf{x}, \boldsymbol{\theta})$
$\mathcal{R}_0^b(\mathbf{x})$	(9.2)	$b(\mathbf{x}, \boldsymbol{\theta})$
$\mathcal{R}_1^b(\mathbf{x})$	(9.3)	$(b(\mathbf{x}, \boldsymbol{\theta}) - \mathcal{R}_0(\mathbf{x})^2$
$\mathcal{R}_2^b(\mathbf{x})$	(9.4)	$\gamma c_1 b(\mathbf{x}, \boldsymbol{\theta}) + (1-\gamma) c_2 (b(\mathbf{x}, \boldsymbol{\theta}) - \mathcal{R}_0(\mathbf{x})^2$
$\mathcal{R}_4^b(\mathbf{x})$	(9.7)	$\mathbf{1}(b(\mathbf{x}, \boldsymbol{\theta}) \geq 0) - (1-\beta)$

Table 14.1: Integrand in Eq. (14.3) for different measures.

For the approximation of Eq. (14.3) at $\mathbf{x}$, we use the MC sample average $\widehat{\mu}_1$ based on N samples $\{\boldsymbol{\theta}_i\}_{i=1}^N$,

$$\mathbb{E}\left[B(\mathbf{x}, \boldsymbol{\theta})\right] = \widehat{\mu}_1\left[B(\mathbf{x}, \boldsymbol{\theta})\right] + \varepsilon = \frac{1}{N}\sum_{i=1}^{N} B(\mathbf{x}, \boldsymbol{\theta}_i) + \varepsilon. \tag{14.4}$$

Here, ε represents the error of the sample approximation. As we have described in Section 6.3 about sampling, we know that $\sqrt{N}\varepsilon$ is asymptotically normally distributed with zero mean and variance $\sigma^2 = \mathbb{V}[B(\mathbf{x}, \boldsymbol{\theta})]$ for $N \to \infty$. This allows to define a confidence interval around the approximated expected value, $\widehat{\mu}_1\left[B(\mathbf{x}, \boldsymbol{\theta})\right]$, which contains $\mathbb{E}\left[B(\mathbf{x}, \boldsymbol{\theta})\right]$ with high probability.

To get a confidence interval

$$[\widehat{\mu}_1\left[B(\mathbf{x}, \boldsymbol{\theta})\right] - \varepsilon,\ \widehat{\mu}_1\left[B(\mathbf{x}, \boldsymbol{\theta})\right] + \varepsilon]$$

that contains $\mathbb{E}\left[B(\mathbf{x}, \boldsymbol{\theta})\right]$ with a probability exceeding $\nu \in]0, 1[$, we compute the sample estimate $\widehat{\sigma}_{\text{biased}}(x)$ of the standard deviation of $\{B(\mathbf{x}, \boldsymbol{\theta}_i)\}_{i=1}^N$ using the unbiased variance estimator,

$$\widehat{\sigma}_{\text{biased}}(x)^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(B(\mathbf{x}, \boldsymbol{\theta}_i) - \widehat{\mu}_1[B(\mathbf{x}, \boldsymbol{\theta})]\right)^2. \tag{14.5}$$

Next, we set

$$\bar{\varepsilon} = \frac{t_\nu\, \widehat{\sigma}_{\text{biased}}(x)}{\sqrt{N}},$$

with t_ν being a constant defining the confidence interval with respect to ν. This constant reflects the Z-score for larger sample sizes. Here, we use $\bar{\varepsilon}$ as an indicator for the upper bound on the sampling error $\varepsilon \leq \bar{\varepsilon}$ with probability exceeding ν. We choose $t_\nu = 2$ in our implementation, which yields a confidence level exceeding 0.975 for a sample size $N \geq 60$.

Given the sampling estimator, we approximate the measures as $\mathcal{R}^b = R^b + \bar{\varepsilon}$, which results in the approximate robust optimization problem that we intent to solve:

$$\begin{aligned} &\min R^f(\mathbf{x}, \boldsymbol{\theta}) \\ \text{s.t.} \quad &R^{c_i}(\mathbf{x}, \boldsymbol{\theta}) \leq 0, i = 1, ..., r. \end{aligned} \tag{14.6}$$

As a remark, we note that our proposed algorithm is not restricted to sample averages as discussed above. For example, we refer to [337, 338, 339] for a sophisticated approximation and error analysis of the robustness measure $\mathcal{R}_3$.

The effectiveness of the NOWPAC algorithm (see Alg. 1) hinges on the accuracy of the surrogate models m_k^b. The surrogate offers us a way to approximate gradients and compute an optimal point, when we solve the trust-region subproblem as given in Eq. (8.8). Therefore, it is crucial to ensure that our algorithm is robust against the noise $\bar{\varepsilon}_x$ that arises from finite sampling approximations of the robustness measures. In the next section, we will discuss how we can adapt the trust-region surrogate to account for both the structural error in the surrogate approximations and the sampling error in the evaluation of $R \approx \mathcal{R}$.

14.2 Noise-adapted trust-region management

The noise in the evaluations of $\mathcal{R}^b = R^b + \bar{\varepsilon}$ impacts the quality of our quadratic surrogate models. Whereas we used minimum Frobenius norm models in the deterministic case of NOWPAC as discussed in Section 8.2, their interpolation condition is disadvantageous if we assume noisy evaluations. Under noise, we would not expect the surrogate points to be interpolated exactly by the model, but rather approximately. This especially is the case when an estimate of the noise is given, as in our case for $\bar{\varepsilon}$.

The work in [173] presents updated minimum Frobenius norm surrogate models, which incorporate the noise and meet error bounds to be fully linear. Instead of requiring an exact interpolation at the surrogate points, the models require only approximate interpolation. Repeating the quadratic surrogate from Eq. (8.5),

$$m_k^b(\mathbf{x}_k + \mathbf{s}) = m_k^b(\mathbf{x}_k) + \mathbf{g}_k\mathbf{s} + \frac{1}{2}\mathbf{s}^T\mathbf{H}_k\mathbf{s} + q_k, \tag{14.7}$$

we find the coefficients of this model as

$$[q_k, \mathbf{g}_k, \mathbf{H}_k] = \underset{q_k, \mathbf{g}_k, \mathbf{H}_k}{\arg\min} ||\mathbf{H}_k - \mathbf{H}_{k-1}||_F^2 : |m_k^b(\mathbf{x}_i) - b(x_i)| < \bar{\varepsilon}, i = 1, ..., k. \tag{14.8}$$

As we observe, instead of the interpolation condition $\{m_k^b(\mathbf{x}_i) = b(x_i)\}_{i=1}^k$, we require only an approximate interpolation with an assumption on the noise, $\bar{\varepsilon}$. As a consequence, depending on this noise estimate, it results in different surrogate models. This approximation tries to prevent overfitting to the data, which can be a result of the exact interpolation condition under noisy assumption. We will showcase this in the following example, where we also compare to a least squares regression model.

Example 10 (Comparison of quadratic models under noise)**.** *We juxtapose the minimum Frobenius norm model with approximate interpolation condition from Eq.* (14.8) *(red dashed) with the minimum Frobenius norm model with exact interpolation condition (green dotted) and a quadratic model found using least-squares regression, i.e.* $[q_k, \mathbf{g}_k, \mathbf{H}_k] = \arg\min_{q_k,\mathbf{g}_k,\mathbf{H}_k} \sum_{i=1}^{k} (m_k^b(\mathbf{x}_i) - b(x_i))^2$ *(black solid). We use five data points given as blue crosses for the surrogate models. The noise estimate, $\bar{\varepsilon}$, for the approximated interpolation condition is given in the title.*

While the quadratic model does not change for the least-squares regression and exact interpolation model for different noise estimates, we observe that including the noise estimate in the approximate interpolation conditions changes the shape of the surrogate model. While for a small noise estimate ($\bar{\varepsilon} = 0.1$) the surrogate model is similar to a quadratic model, for a slightly larger noise estimates ($\bar{\varepsilon} = 0.2$), the surrogate is almost linear. Thus, by including the noise as parameter, we observe that a linear function might suffice as approximation, whereas least-squares regression and the exact interpolation condition results in a quadratic model. Hence, we avoid overfitting to the data.

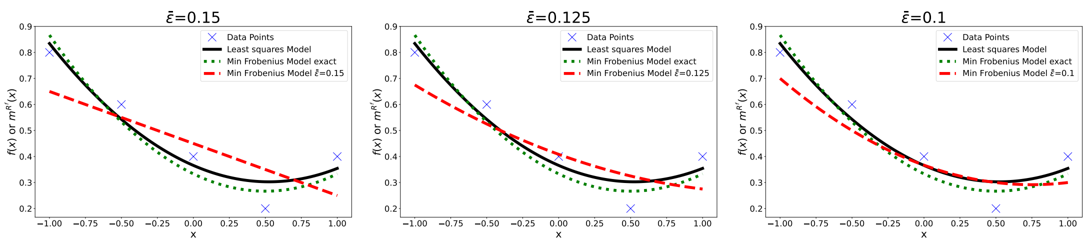


Figure 14.1: Comparison of quadratic models using least squares, minimum Frobenius norm with exact and approximated interpolation condition for different $\bar{\varepsilon}$.

Given these surrogate models built from finite sample approximations with black-box evaluations corrupted by noise, it is established in [173, Thm. 2.2] that they preserve the fully-linear condition and meet the following error bounds,

$$\begin{aligned} \left\| \mathcal{R}^b(x_k + s) - m_k^{R^b}(x_k + s) \right\| &\leq \kappa_1\, \rho_k^2, \\ \left\| \nabla \mathcal{R}^b(x_k + s) - \nabla m_k^{R^b}(x_k + s) \right\| &\leq \kappa_2\, \rho_k, \end{aligned} \tag{14.9}$$

with high probability ν.

The constants in these bounds,

$$\kappa_i = \kappa_i \left(\bar{\varepsilon}_{\max}^k \rho_k^{-2} \right), \qquad i \in \{1, 2\}, \tag{14.10}$$

depend on the statistical upper bounds estimates for the noise term, $\bar{\varepsilon}_{\max}^k = \max\{\bar{\varepsilon}_i\}_{i=1}^k$, as presented in Section 14.1, the trust-region radius ρ_k as well as on the poisedness constant $\Lambda \geq 1$ (see Remark 2).

We note two opposing properties in this bound: to improve the quality of the surrogate models, we need to reduce the poisedness constant Λ of the points used to build the regularized surrogate models. The bound also improves with a smaller trust-region radius ρ_k. At the same time, however, we have to ensure boundedness of the error term $\bar{\varepsilon}^k_{\max}\rho_k^{-2}$, which grows with a decreasing trust-region size. We will discuss the balance of both measures next.

Firstly, regarding the poisedness constant, we decrease its value Λ in every rejected step by a user-prescribed factor $\gamma_{\text{poi}} \in]0,1]$ (down to $\Lambda_{\min}$). This helps to increase the quality of the surrogate models until the trust region becomes small enough and the noise term $\bar{\varepsilon}^k_{\max}\rho_k^{-2}$ in (14.10) starts dominating the poisedness constant Λ. We refer to Alg. 3 for pseudo-code regarding the implementation.

Algorithm 3 Updating procedure for poisedness threshold

```
if Current trial step rejected then
    Reduce poisedness constant of interpolation points
    Set $\Lambda = \max\{\gamma_{\text{poi}}\,\Lambda,\ \Lambda_{\min}\}$
end if
if Current trial step successful then
    Relax poisedness constant of interpolation points
    Set $\Lambda = \min\{\Lambda/\gamma_{\text{poi}},\ \Lambda_0\}$
end if
Update surrogate models $m_k^f$ and $m_k^c$
```

Secondly, if the maximal noise term $\bar{\varepsilon}^k_{\max}$ is of order ρ_k^2, the bounds in (14.9) hold. However, in the presence of noise, i.e. $\bar{\varepsilon}^k_{\max} > 0$, the term $\bar{\varepsilon}^k_{\max}\rho_k^{-2}$ grows unboundedly for a shrinking trust-region radius, and consequently, κ_1 and κ_2 also increase unboundedly, violating the fully-linearity property of $m_k^{R^b}$. Therefore, to ensure the full linearity of the surrogate models, it is necessary to impose an upper bound on the error term, $\bar{\varepsilon}^k_{\max}\rho_k^{-2}$. Our algorithm enforces the lower bound

$$\bar{\varepsilon}^k_{\max}\rho_k^{-2} \le \lambda_t^{-2}, \quad \text{resp.} \qquad \rho_k \ge \lambda_t\sqrt{\bar{\varepsilon}^k_{\max}}, \tag{14.11}$$

on the trust-region radius for a $\lambda_t \in]0,\infty[$, where we set $\lambda_t = \sqrt{2}$ by default.

Algorithm 4 specifies the details of the implementation. Notice the difference to the trust-region management in NOWPAC (see Alg. 1), where the trust region is only modified in the acceptance step.

Algorithm 4 Noise adapted updating procedure for trust-region radius.

```
Input: trust-region factor $a \in \{1, \gamma_{\text{dec}}, \gamma_{\text{inc}}, \omega\}$, maximum trust region $\rho_{\max}$.
Set $\rho_{k+1} = \max\left\{a\rho_k,\ \lambda_t\sqrt{\bar{\varepsilon}^k_{\max}}\right\}$
if $\rho_{k+1} > \rho_{\max}$ then
    Set $\rho_{k+1} = \rho_{\max}$
end if
```

The proposed noise-adapted trust-region management approach takes into account both the structural error of the fully linear approximation and the highly probable upper bound on the error in the approximation of the robustness measures. This coupling, however, prevents the trust-region radius from converging to zero, which results in limited accuracy of the surrogate models $m_k^{R^b}$ and consequently, the optimization result. While this approach effectively limits the impact of noise on the surrogate models, it also restricts the level of accuracy that can be achieved; in other words, there is a trade-off between the level of robustness to noise and the accuracy of the surrogate models. In the next section, we will discuss a measure to reduce the noise itself to be able to reduce the trust-region radius: GP surrogates.

14.3 Noise correction supported by Gaussian processes

We can improve the poisedness constant, and thus the surrogate, mainly through selecting points used for the surrogate; however, this is restricted by the evaluations available. The alternative is to reduce the magnitude of the noise term $\bar{\varepsilon}_{\max}^k$ in order to increase the accuracy of the optimization result.

We know from Section 6.3, that the MC sampling variance is $\frac{\sigma^2}{N}$. A straight-forward solution would, thus, be to increase the number of samples N to decrease the noise. However, given that increasing the number of samples is prohibitively expensive since the sampling error only decreases in the order of $\mathcal{O}(\frac{1}{\sqrt{N}})$, this is infeasible. Hence, we propose an alternative approach:

We introduce GP surrogates of $\mathcal{R}^b$, as presented in Chap. 7, by training the surrogate on previously evaluated optimization points $\{(\mathbf{x}_i, R_i^b)\}_{i=1}^K$ in a larger domain encompassing the trust region. By incorporating this second surrogate, we can decrease the error and enhance the estimator's smoothness. We leverage the GP's global information, which improves as the number of points increases. We take advantage of the GP's consistency properties and smooth behavior, as demonstrated in [280, 299, 319], most recently in [198] and summarized in Section 7.5. In the following sections, we are going to show how this approach helps us smooth the noisy evaluations and reduce the magnitude of the noise term $\bar{\varepsilon}_{\max}^k$.

14.3.1 Gaussian process construction

We refer to the information from Chap. 7 on GPs and repeat the GP estimators for mean and variance, extending them to the notation of our algorithm for a general training data $(\mathbf{X}, \mathbf{y})$ as

$$\text{Mean: } \mathcal{G}_k^b[\mathbf{y}] := \mathcal{G}^b[\mathbf{x}_k|\mathbf{X}, \mathbf{y}] = \mathbf{k}_{\mathbf{x}_k\mathbf{X}}(\mathbf{K}_{\mathbf{XX}}^b + \mathbf{N})^{-1}\mathbf{y}, \tag{14.12}$$

$$\text{Variance: } \mathcal{V}_k^b := \mathcal{V}[\mathbf{x}_k|\mathbf{X}] = \mathbf{k}_{\mathbf{x}_k\mathbf{x}_k}^b - \mathbf{k}_{\mathbf{x}_k\mathbf{X}}^b(\mathbf{K}_{\mathbf{XX}}^b + \mathbf{N})^{-1}\mathbf{k}_{\mathbf{X}\mathbf{x}_k}^b, \tag{14.13}$$

where we introduce the target function $b \in \{f, c_1, ..., c_r\}$ and the current optimization iteration k.

Here, for the training data, $\mathbf{X}$ represents our already evaluated design points $\{\mathbf{x}_i\}_{i=1}^k$ and $\mathbf{R}^b$ denotes the corresponding evaluations of the measure $\{R_i^b\}_{i=1}^k$. The symbols $\mathbf{k}^b_{\mathbf{x}_k\mathbf{X}}$ and $\mathbf{K}^b_{\mathbf{XX}}$ denote the kernel vector and kernel matrix, respectively, evaluated at every pair $(\mathbf{x}, \mathbf{x}'), \mathbf{x}, \mathbf{x}' \in X$. The noise matrix $\mathbf{N}$ denotes a diagonal matrix with our noise estimates $\bar{\varepsilon}_k^b$ coming from the evaluation of the robustness measures on its diagonal. In what follows, we use the shorthand notation $\mathcal{G}_k^b := \mathcal{G}_k^b[\mathbf{y}]$ when the training set is clear from the context to simplify notation. We note, that we build an independent GP for each of the objective and constraint functions, $b \in \{f, c_1, ..., c_r\}$.

We consider already evaluated points with a distance smaller than $\zeta_1\rho_k$ around the current best design point $\mathbf{x}_k$ for the construction of the GPs to incorporate more global information compared to the local surrogates, i.e.

$$(\mathbf{X}, \mathbf{R}^b) = \{(\mathbf{x}_j, R_j^b) : \|\mathbf{x}_j - \mathbf{x}_k\|_2 \leq \zeta_1\rho_k, j = 1, ..., K\}. \tag{14.14}$$

While this includes more points than for the local surrogate, we can still assume stationarity of the GP surrogates, since we still restrict ourselves to a local domain around the current optimal point. Of course, this specifically requires the current point of evaluation $\mathbf{x}_k$ being included in the training set. By default, we use a value of $\zeta_1 = 3$ to incorporate enough global information around the current design.

14.3.2 Gaussian process smoothing

To enhance the accuracy of the finite sample approximation R_k^b, we need to reduce its error. If the estimator is unbiased, this results in reducing the variance as we showed in Section 6.3. We achieve this by incorporating the GP surrogate estimates $\mathcal{G}_k^b[\mathbf{R}]$ as a second estimator, which gets more and more precise as the number of evaluations $\mathbf{R}$ increases during the optimization process. We combine the two estimators using a linear combination and adjust their contributions with a weighting factor γ_k. The resulting *smoothened* evaluation is denoted as $\tilde{R}_k^b$ and can be expressed as:

$$\tilde{R}_k^b = \gamma_k \mathcal{G}_k^b[R_k^b] + (1 - \gamma_k)R_k^b, \tag{14.15}$$

where R_k^b represents the noisy sampling estimate, and $\mathcal{G}_k^b[R_k^b]$ is the mean estimator of the GP at the current evaluation $\mathbf{x}_k$.

However, what is the best choice for γ_k? Can we find the optimal γ_k to get the best estimator $\tilde{R}_k^b$? We developed two options to choose γ_k in SNOWPAC: analytic and heuristic smoothing.

Analytic smoothing

Under the valid assumption that R_k^b is itself an unbiased estimator, we compute the root mean squared error (RMSE) of $\tilde{R}_k^b$ as

$$\begin{aligned} \mathrm{RMSE}(\tilde{R}_k^b) &= [\gamma_k \mathbb{E}[\mathcal{G}_k^b[R_k^b] - R_k^b]]^2 + \mathbb{V}[\gamma_k \mathcal{G}_k^b[R_k^b] + (1-\gamma)R_k^b] \\ &= [\gamma_k(\mathcal{G}_k^b[\mathcal{R}_k^b] - \mathcal{R}_k^b)]^2 + \gamma_k^2 \mathbb{V}[\mathcal{G}_k^b[R_k^b]] + (1-\gamma_k)^2 \mathbb{V}[R_k^b] \\ &+ 2\gamma_k(1-\gamma_k)\mathbb{C}\mathrm{ov}[\mathcal{G}_k^b[R_k^b], R_k^b]. \end{aligned} \tag{14.16}$$

The goal is to find the optimal $\overset{*}{\gamma}_k$ to minimize (14.16) of the new estimator $\tilde{R}_k^b$. By doing so, we can adjust its noise estimate

$$\tilde{\varepsilon}_k^b = t_\nu \cdot \min_{\gamma_k} \mathrm{RMSE}(\tilde{R}_k^b). \tag{14.17}$$

Lemma 20. *The optimal $\overset{*}{\gamma}_k$ that minimizes $\arg\min_{\gamma_k}$ RMSE$(\tilde{R}_k^b)$ is given as*

$$\overset{*}{\gamma}_k = \frac{\mathbb{V}[R_k^b] - \mathbb{C}ov[\mathcal{G}_k^b[R_k^b], R_k^b]}{(\mathcal{G}_k^b[\mathcal{R}_k^b] - \mathcal{R}_k^b)^2 + \mathbb{V}[\mathcal{G}_k^b[R_k^b]] + \mathbb{V}[R_k^b] - 2\mathbb{C}ov[\mathcal{G}_k^b[R_k^b], R_k^b]}. \tag{14.18}$$

Proof. Taking the derivative of (14.16) for γ_k and setting it equal to 0 gives us its optimal value to minimize the error. □

As a final step, we describe how to compute all the terms in (14.16) and (14.18). We can compute them in closed form since the GP mean operator is linear.

Lemma 21. *The variance of the GP mean estimator $\mathcal{G}_k^b[R_k^b]$ is given as:*

$$\begin{aligned}\mathbb{V}[\mathcal{G}_k^b[R_k^b]] &= \sum_{i=1}^{N} \mathbb{V}[R_i^b]\Big(\sum_{j=1}^{N} \mathbf{k}_{\mathbf{x}_k\mathbf{x}_j}((K_{\mathbf{XX}} + \mathbf{N})^{-1})_{[i,j]}\Big)^2 \\ &\approx \sum_{i=1}^{N} \Big(\frac{\bar{\varepsilon}_{\mathbf{i}}^{\mathbf{b}}}{t_\nu}\Big)^2 \Big(\sum_{j=1}^{N} \mathbf{k}_{\mathbf{x}_k\mathbf{x}_j}((K_{\mathbf{XX}} + \mathbf{N})^{-1})_{[i,j]}\Big)^2\end{aligned} \tag{14.19}$$

See D.2 for the proof.

Here, the notation $\mathbf{X}_{[i,j]}$ denotes the element of $\mathbf{X}$ at position $[i,j]$. We want to make the distinction clear that we compute the variance of the GP mean estimator $\mathbb{V}[\mathcal{G}_k^b[R_k^b]]$ in (14.19) which is not the same as the variance estimate $\mathcal{V}_k^b$ of the GP surrogate.

Next, we have to compute the covariance term.

Lemma 22. *The covariance between the GP mean estimator $\mathcal{G}_k^b[R_k^b]$ and sampling estimator R_k^b is given as:*

$$\begin{aligned}\mathbb{C}ov[\mathcal{G}_k^b[R_k^b], R_k^b] &= \mathbb{V}[R_k^b] \sum_{j=1}^{N} \mathbf{k}_{\mathbf{x}_k\mathbf{x}_j}((K_{\mathbf{XX}} + \mathbf{N})^{-1})_{[k,j]} \\ &\approx \Big(\frac{\bar{\varepsilon}_k^b}{t_\nu}\Big)^2 \sum_{j=1}^{N} \mathbf{k}_{\mathbf{x}_k\mathbf{x}_j}((K_{\mathbf{XX}} + \mathbf{N})^{-1})_{[k,j]}.\end{aligned} \tag{14.20}$$

See D.3 for the proof.

Here, too, we compute the covariance between two estimators, namely $\mathcal{G}_k^b[R_k^b]$ and R_k^b in (14.20) and not between evaluations themselves.

The last term in (14.16) and (14.18) that we still require is $(\mathcal{G}_k^b[\mathcal{R}_k^b] - \mathcal{R}_k^b)$. We note that this term is the bias of $\mathcal{G}_k^b$:

$$\mathrm{Bias}(\mathcal{G}_k^b) = \mathbb{E}[\mathcal{G}_k^b(\mathbf{x}_k^{(i)}; \mathbf{R})] - \mathcal{R}_k^b = \mathcal{G}_k^b(\mathbf{x}_k^{(i)}; \mathbb{E}[\mathbf{R}]) - \mathcal{R}_k^b = \mathcal{G}_k^b[\mathbf{x}_k^{(i)}; \mathcal{R}] - \mathcal{R}_k^b. \tag{14.21}$$

We can approximate the bias term using a Bootstrap approach (cf. [106, 107]).

Remark 3. ***Estimating the bias through bootstrapping.*** *Let $X \sim F$ be a random variable whose cumulative distribution function is given by $F(x)$. We denote a parameter θ as a function of the distribution F, denoted as $\theta = T(F)$ or θ_F for short. Examples for θ include the mean or standard deviation. Let $X_1, X_2, \ldots, X_N$ be N random samples of X from the estimated cumulative distribution function $\hat{F}$. An estimator $\hat{\theta}$, is a function of the sample, i.e. $\hat{\theta} = h(X_1, X_2, \ldots, X_N)$. For example, if $\theta_F = \mu_F$, then $\hat{\theta}_{\hat{F}} = \frac{1}{N}\sum_{i=1}^{N} X_i$ is an estimator for the mean.*

We define the bias of the estimator $\hat{\theta}_{\hat{F}}$ as

$$Bias_F(\hat{\theta}_{\hat{F}}) = \mathbb{E}_F[\hat{\theta}_{\hat{F}}] - \theta_F. \tag{14.22}$$

The bootstrap estimate of the bias is given as [107]

$$Bias_{\hat{F}}(\hat{\theta}_{\hat{F}}) = \mathbb{E}_{\hat{F}}[\hat{\theta}_{\hat{F}}] - \hat{\theta}_{\hat{F}}. \tag{14.23}$$

The first term is approximated using the Bootstrap approach by resampling from $\hat{F}$ with replacement, i.e. we draw N samples from our set of samples $\{X^{(i)}\}_{i=1}^{N}$ and we repeat this process S times. Let $\{\hat{X}_s^{(i)}\}_{i=1}^{N}, s = 1, ..., S$, denote a set of these resampled samples. We can then approximate the term as

$$\mathbb{E}_{\hat{F}}[\hat{\theta}_{\hat{F}}] \approx \frac{1}{S}\sum_{s=1}^{S} \hat{\theta}(\hat{X}_s). \tag{14.24}$$

In our case, we denote $\mathbb{E}_{\hat{F}}[\hat{\theta}_{\hat{F}}] := \mathbb{E}[\mathcal{G}_k^b(\mathbf{x}_k^{(i)}; \mathbf{R})] \approx \mathbb{E}[\mathcal{G}_k^b[\mathbf{x}_k^{(i)}; \hat{\mathbf{R}}]]$ as the Bootstrap estimator for the first term where we draw S samples $\hat{\mathbf{R}}$ with replacement. For the second term we have $\theta_F := \mathcal{R}_k^b$ and since $\mathcal{G}_k^b(\mathbf{x}_k^{(i)}; \mathbf{R})$ estimates $\mathcal{R}_k^b$ we get $\hat{\theta}_{\hat{F}} := \mathcal{G}_k^b(\mathbf{x}_k^{(i)}; \mathbf{R})$. Thus, we approximate (14.21) by

$$\mathcal{G}_k^b[\mathcal{R}] - \mathcal{R}_k^b \approx \mathbb{E}[\mathcal{G}_k^b[\hat{\mathbf{R}}]] - \mathcal{G}_k^b. \tag{14.25}$$

With the derivations in Lemmas 21 and 22 and the Bootstrap approximation of Eq. (14.25), we can compute $\overset{*}{\gamma}_k$, which results in the minimal RMSE for our smoothend estimator $\tilde{R}_k^b$. The accuracy of the mentioned quantities is highly reliant on both the GP's approximation quality and the robustness measures R^b. The first improves with more available evaluations, i.e. optimization steps. The second mainly depends on the problem and the number of samples N. As a compromise, we offer a heuristic method for computing γ_k rather than calculating $\overset{*}{\gamma}_k$, which is especially useful for small numbers of samples or optimization steps.

Heuristic smoothing

We reduce the noise by applying a linear combination comparable to the one presented in (14.15), and utilizing the GP variance estimator:

$$\tilde{\varepsilon}_k^b \;=\; \gamma_k t_\nu \sqrt{\mathcal{V}_k^b} + (1-\gamma_k)\bar{\varepsilon}_k^b. \tag{14.26}$$

The weight factor $\gamma_k := e^{-\sqrt{\mathcal{V}_k^b}}$ is chosen to approach 1 when the GP becomes more and more accurate as indicated by the vanishing variance of the GP approximation.

The intention behind both the analytic and the heuristic error is to exploit posterior consistency of the GP for $\mathcal{V}_k^b$ converging to zero for an increasing number of evaluations of R^b within a neighborhood of $\mathbf{x}_k$. In the limit, the GP mean converges to the exact function $\mathcal{R}^b$ and the lower bound of Eq. (14.11) on the trust-region radius vanishes, allowing for increasingly accurate optimization results.

14.3.3 Gaussian process error balancing

By utilizing both the fully linear and GP surrogate models, we can balance two sources of approximation errors. On the one hand, there is a structural error in the approximation of the local surrogate models, as shown in Eq. (14.9), which can be managed by controlling the size of the trust-region radius. On the other hand, we have the inaccuracy in the GP surrogate itself, which is indicated by the variance $\mathcal{V}_k^b$ of the GP. It is worth noting that Algorithm 4 connects these two sources of errors by associating the size of the trust-region radius with the size of the credible interval through Eq. (14.15). The algorithm only permits the trust-region radius to decrease if $\mathcal{V}_k^b$ becomes small.

Finally, we take three measures to ensure that $\mathcal{V}_k^b$ reduces as $\mathbf{x}_k$ gets closer to the optimal design. Firstly, the number of black-box evaluations performed by the optimizer during the optimization process naturally increases, which improves the quality of the GP approximation approaching $\mathcal{R}_k^b$ for an increasing number of evaluations (i.e. training data), see [198, 280, 299, 319] and Section 7.5 of this thesis. However, these evaluations may not be well-distributed around the current iterate $\mathbf{x}_k$, and may be localized.

Therefore, secondly, we draw additional points, $\hat{\mathbf{x}} \sim \mathcal{N}\left(\mathbf{x}_k, \zeta_2\sqrt{\rho_k} I\right)$, with $\zeta_2 = \frac{3}{10}$, whenever a trial point is rejected to improve the geometrical distribution of the regression points for the GP surrogates. A trial point being rejected may happen because it is infeasible under the current GP-corrected constraint approximation in Eq. (14.15), or the step is rejected (as in `STEP 3` in Algorithm 1).

Thirdly, in addition to enriching the set of regression points, we re-estimate the GP hyperparameters either after a user-prescribed number of black-box evaluations or after $\lambda_k \cdot n$ consecutively rejected or infeasible trial steps. Here, λ_k is a user-prescribed constant. This avoids over-fitting problems as described in Chap. 7 or [62, 269].

14.4 Relaxed feasibility requirement

One of the essential components of Algorithm 1 involves ensuring the feasibility of all intermediate design points $\mathbf{x}_k$ through the feasibility requirement outlined in `STEP 2`.

Nevertheless, checking feasibility in the presence of noise can pose significant challenges. For instance, we may accept a seemingly feasible point based on the current constraint approximations, which may actually be infeasible due to the noise. Additionally, results might become infeasible (or feasible) due to the smoothening step of $\{\tilde{R}^b_{c_i}\}_{i=1}^r$ in Eq. (14.15). Hence, it becomes necessary to expand NOWPAC's abilities to handle infeasible points by introducing a feasibility-restoration mode.

Consequently, the resulting algorithm operates in two modes—M1 and M2:

(M1) objective minimization and

(M2) feasibility restoration.

Whenever the current point $\mathbf{x}_k$ seems feasible based on the current constraint approximations, the algorithm operates in mode (M1). However, if $\mathbf{x}_k$ becomes infeasible, the algorithm switches to mode (M2). We present the switch in Alg. 5.

Algorithm 5 Switch between normal and feasibility restoration mode.

```
1: **Input**: Design $\mathbf{x}$.
2: **if** $\tilde{R}^{c_i}(\mathbf{x}) \leq 0 \; \forall \; i = 1, ..., r$ **then**
3:     Switch to mode (M1).
4: **else**
5:     Switch to mode (M2).
6: **end if**
```

The switch between these two modes involves replacing the underlying trust-region subproblem. In mode (M1), the standard subproblem

$$\begin{aligned} &\min m_k^{\tilde{R}^f}(\mathbf{x}_k + \mathbf{s}_k), \\ \text{s.t.} \quad &\bar{m}_k^{\tilde{R}^{c_i}}(\mathbf{x}_k + \mathbf{s}_k) \leq 0, \; i = 1 \dots r \\ &\|\mathbf{s}_k\| \leq \rho_k \end{aligned} \tag{14.27}$$

is solved to obtain a new trial point $\mathbf{x}_k + \mathbf{s}_k$. Here, $\bar{m}_c^{\tilde{R}^{c_i}}$ denotes the inner-boundary path augmented models of $\mathcal{R}^{c_i}$ as described in (8.9) using the updated evaluations $\tilde{R}^b_k$ from GP smoothening in Eq. (14.15). The subproblem

$$\begin{aligned} &\min \left\langle \mathbf{g}_k^{\tilde{R}^f}, \mathbf{s}_k \right\rangle, \\ \text{s.t.} \quad &\bar{m}_k^{\tilde{R}^{c_i}}(\mathbf{x}_k + \mathbf{s}_k) \leq 0, \; i = 1 \dots r \\ &\|\mathbf{s}_k\| \leq \rho_k \end{aligned} \tag{14.28}$$

is used for computation of the criticality measure α_k.

In mode (M2), we minimize the constraint violation instead of optimizing the objective, moving back into the feasible region. Thus, the subproblem

$$\begin{aligned} \min \sum_{i \in \mathcal{I}_k} & \left(m_k^{\tilde{R}^{c_i}}(\mathbf{x}_k + \mathbf{s}_k)^2 + \lambda_g m_k^{\tilde{R}^{c_i}}(\mathbf{x}_k + \mathbf{s}_k) \right), \\ \text{s.t.} \quad & \bar{m}_k^{\tilde{R}^{c_i}}(\mathbf{x}_k + \mathbf{s}_k) \leq \tau_i, \; i = 1 \dots r \\ & \|\mathbf{s}_k\| \leq \rho_k \end{aligned} \tag{14.29}$$

is solved for the computation of a new trial point $\mathbf{x}_k + \mathbf{s}_k$, along with

$$\begin{aligned} \min \sum_{i \in \mathcal{I}_k} & \left(2 m_k^{\tilde{R}^{c_i}}(\mathbf{x}_k) + \lambda_g \right) \left\langle \mathbf{g}_k^{m_k^{\tilde{R}^{c_i}}}, \mathbf{s}_k \right\rangle, \\ \text{s.t.} \quad & \bar{m}_k^{\tilde{R}^{c_i}}(\mathbf{x}_k + \mathbf{s}_k) \leq \tau_i, \; i = 1 \dots r \\ & \|\mathbf{s}_k\| \leq \rho_k \end{aligned} \tag{14.30}$$

for computation of the corresponding criticality measure. We denote the set of violated constraints as $\mathcal{I}_k = \{i \; : \; \tilde{R}_k^{c_i} > 0, \; i = 1, \dots, r\}$ and introduce the slack variables $\tau := (\tau_1, \dots, \tau_r)$, which we set as $\tau_i = \max\{\tilde{R}_k^{c_i}, \; 0\}$. The parameter $\lambda_g \geq 0$ in (14.29) and (14.30) is introduced to guide the feasibility restoration towards the interior of the feasible domain. By default it is set to $\lambda_g = 10^{-4}$.

The respective mode also affects the acceptance ratio r_k. While we compute the usual acceptance ratio in (M1), we adapt the terms to the formulations to the changed objective of Eq. 14.29 in (M2). We compare the performance of the surrogate to the sample measures on the violated constraints. Therefore, we adapt the algorithm as shown in Alg. 6.

Algorithm 6 Compute acceptance ratio r_k in normal and feasibility restoration mode.

```
1: Input: Current design $\mathbf{x}_k$ and trial point $\mathbf{x}_{\text{trial}} = \mathbf{x}_k + \mathbf{s}_k$.
2: if Mode (M1) then
3:     $r_k = \frac{\tilde{R}^f(\mathbf{x}_k) - \tilde{R}^f(\mathbf{x}_{\text{trial}})}{m_k^{\tilde{R}^f}(\mathbf{x}_k) - m_k^{\tilde{R}^f}(\mathbf{x}_{\text{trial}})}$.
4: else
5:     $r_k = \frac{\sum_{i \in \mathcal{I}_k} \left( \tilde{R}^{c_i}(\mathbf{x}_k)^2 + \lambda_g \tilde{R}^{c_i}(\mathbf{x}_k) \right) - \sum_{i \in \mathcal{I}_k} \left( \tilde{R}^{c_i}(\mathbf{x}_{\text{trial}})^2 + \lambda_g \tilde{R}^{c_i}(\mathbf{x}_{\text{trial}}) \right)}{\sum_{i \in \mathcal{I}_k} \left( m_k^{\tilde{R}^{c_i}}(\mathbf{x}_k)^2 + \lambda_g m_k^{\tilde{R}^{c_i}}(\mathbf{x}_k) \right) - \sum_{i \in \mathcal{I}_k} \left( m_k^{\tilde{R}^{c_i}}(\mathbf{x}_{\text{trial}})^2 + \lambda_g m_k^{\tilde{R}^{c_i}}(\mathbf{x}_{\text{trial}}) \right)}$.
6: end if
```

14.5 Summary of the extensions to NOWPAC

Before we present the new algorithm, we summarize the changes to NOWPAC that are required for OUU. As basic assumption for these changes, we expected the involved measures are approximated and an estimation of the approximation error is available. In our context, we considered sampling estimators for the statistical estimation (Section 14.1).

These approximated measures and the resulting noise required an updated trust-region management. Firstly, we updated the surrogate models such that the fully-linear property is still fulfilled, which introduced a lower bound on our trust-region radius (Section 14.2). Secondly, to still be able to reduce the trust-region radius, we needed to reduce the noise. Thus, we suggested GP models as second surrogates (Section 14.3). Using these surrogates, we were able to smoothen the evaluations presenting an analytic and heuristic approach (Section 14.3.2 and 14.3.2) and balance the error of the sampling and the GP approximation (Section 14.3.3). Thirdly and finally, the noisy evaluations and the smoothing step resulted in infeasible iterates, which we had not allowed in NOWPAC. Hence, we presented a feasibility-restoration mode, where we change the optimization problem guiding the optimizer back into the feasible region (Section 14.4).

15 The stochastic trust-region algorithm SNOWPAC

In this section, we present the algorithm SNOWPAC (stochastic NOWPAC). We start off by summarizing all the default values for the internal parameters used in our implementation in Table 15.1.

Table 15.1: Default values for internal parameters of SNOWPAC.

Description	Parameter	Default value
Factor for lower bound on trust-region radii	λ_t	$\sqrt{2}$
Poisedness threshold	Λ	100
Gradient contribution to feasibility restoration	λ_g	10^{-4}
Factor for GP training region	ζ_1	3
Constant for normal distribution to enrich GP	ζ_2	$\frac{3}{10}$

SNOWPAC is written in `C++` and is available on Github under the BSD 2-Clause license[1]. It can be used as a stand-alone solver. The installation requires `cmake` and can be started by using the included `install_nowpac.sh` script. In the script, the user can modify the compilers and the installation path. The different possible settings, options and install instructions of SNOWPAC are also available in the `README.md` file in the repository.

For coupling SNOWPAC to the user's application, the `BlackBoxBaseClass.hpp` serves as interface to SNOWPAC. The class offers two methods called `evaluate`, which are responsible for the black-box evaluation. The two methods only differ in their number of input parameters. The stochastic method for SNOWPAC requires a `noise` vector, whereas the deterministic evaluate method for NOWPAC does not. The application has to inherit from `BlackBoxBaseClass.hpp`, link to SNOWPAC and implement the `evaluate` method, which is called by SNOWPAC for evaluations of objective and constraints. Thus, the random sampling and evaluation of measures is handled on the application side. In the case of SNOWPAC, the `evaluate` method also returns noise estimates for all functions, which are used for $\bar{\varepsilon}$. We visualize the coupling in Fig. 15.1.

SNOWPAC is also available as an external solver in Dakota. It can be installed using the option `HAVE_NOWPAC=True` when building DAKOTA from source and chosen by setting `method` to `snowpac`. The settings of SNOWPAC are set through the Dakota input

[1]`https://github.com/snowpac/snowpac`

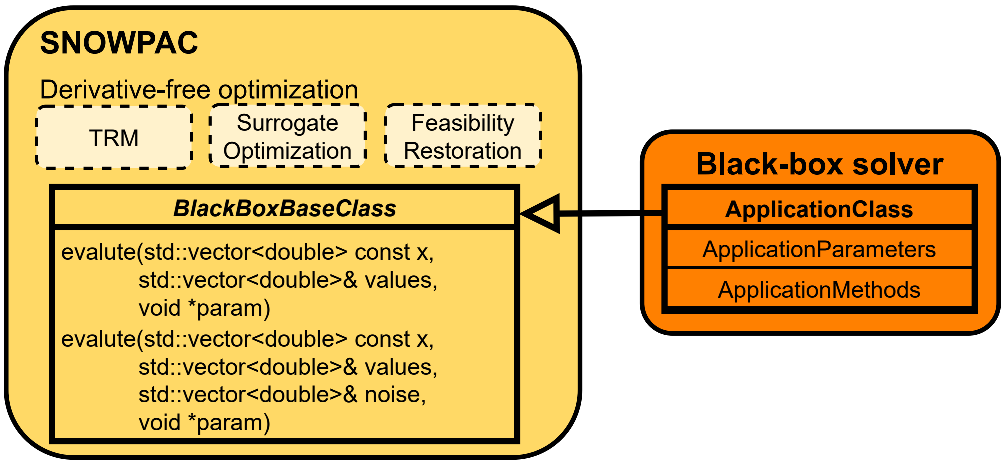

SNOWPAC
Derivative-free optimization
TRM
Surrogate Optimization
Feasibility Restoration
BlackBoxBaseClass
evalute(std::vector<double> const x, std::vector<double>& values, void *param)
evalute(std::vector<double> const x, std::vector<double>& values, std::vector<double>& noise, void *param)
Black-box solver
ApplicationClass
ApplicationParameters
ApplicationMethods

Figure 15.1: Coupling of SNOWPAC with black-box application via `BlackBoxBaseClass.hpp`. We show also the new contributions of SNOWPAC: the updated trust-region management (TRM), surrogate optimization with GPs and feasibility restoration mode.

file, e.g., the initial radius of the trust region or the maximum number of function evaluations. Dakota is then responsible for handling the communication between SNOWPAC and the black-box application. Here, Dakota is also responsible for the sampling of stochastic input variables and evaluation of measures. It makes use of its `nested_model`, where SNOWPAC can be defined as outer optimization method, whereas a sampling method for a SoI is chosen as inner method. In the following, we give an example of a setting for the setup of SNOWPAC in Lst 15.1, where all the different settings of SNOWPAC are specified.

```
method,
    id_method = 'OPTIM'
    model_pointer = 'OPTIM_M'
    snowpac
      seed = 25041981
      max_function_evaluations = 100
      trust_region
        initial_size = 0.05
        minimum_size = 1.0e-6
        contract_threshold = 0.25
        expand_threshold   = 0.75
        contraction_factor = 0.50
        expansion_factor   = 1.50
```

Listing 15.1: Extract of Dakota input file for SNOWPAC.

Lastly, we present the algorithm for SNOWPAC in Alg. 4. The algorithm's general procedure closely follows the steps outlined in Algorithm 1 of NOWPAC. Additionally, it incorporates the generalizations we introduced in Sections 14.2-14.4 to handle noisy black-box evaluations.

Algorithm 7 SNOWPAC

```
1: Construct the initial fully linear models $m_0^{R^f}$, $m_0^{R^c}$.
2: Set $\overset{*}{\mathbf{x}} = \mathbf{x}_0$ and $\overset{*}{R}^f = R^f(\mathbf{x}_0)$.
3: Switch Mode if feasibility restoration is necessary as in Alg. 5.
4: for $k = 0, 1, \dots, n_{max}$ do
5: // STEP 0: Criticality step
6:   while $\alpha_k(\rho_k)$ is too small do
7:     if $\rho_k < \rho_{min}$ then
8:       Return $\overset{*}{\mathbf{x}} = \mathbf{x}_k$, $\overset{*}{R}^f = R^f(\mathbf{x}_k)$. STOP.
9:     end if
10:    Call Algorithm 4 with $a = \omega$.
11:    Evaluate $R^f(\mathbf{x})$ and $R^c(\mathbf{x})$ for randomly sampled $\mathbf{x} \in B(\mathbf{x}_k, \rho_k)$.
12:    Update GPs and black-box evaluations.
13:    Construct surrogate models $m_k^{\tilde{R}^f}$, $m_k^{\tilde{R}^c}$.
14:    Switch Mode according to observed feasibility of trial point as in Alg. 5.
15:  end while
16: // STEP 1: Step calculation
17:  Compute a trial step $\mathbf{x}_{\text{trial}} = \mathbf{x}_k + \mathbf{s}_k$ via (14.27) or (14.29).
18:  Evaluate $R^f(\mathbf{x}_{\text{trial}})$ and $R^c(\mathbf{x}_{\text{trial}})$.
19:  Update Gaussian processes and black box evaluations.
20: // STEP 2: Check feasibility of trial point
21:  if $R^{c_i}(\mathbf{x}_{\text{trial}}) > \tau_i$ for an $i = 1, \dots, r$ then
22:    Call Algorithm 4 with $a = \omega$.
23:    Evaluate $R^f(\mathbf{x})$ and $R^c(\mathbf{x})$ for a randomly sampled $\mathbf{x} \in B(\mathbf{x}_k, \rho_k)$.
24:    Update GPs, black-box evaluations and surrogate models.
25:    Switch Mode to observed feasibility of trial point as in Alg. 5.
26:  end if
27: // STEP 3: Acceptance of trial point and update trust region
28:  Compute $r_k$ according to Algorithm 6.
29:  if $r_k > \eta_0$ then
30:    Set $\mathbf{x}_{k+1} = \mathbf{x}_{\text{trial}}$
31:    Call Algorithm 4 with $a = \min\{1, \gamma_{inc}\}$
32:  else
33:    Set $\mathbf{x}_{k+1} = \mathbf{x}_k$, $m_{k+1}^{\tilde{R}^f} = m_k^{\tilde{R}^f}$ and $m_{k+1}^{\tilde{R}^c} = m_k^{\tilde{R}^c}$
34:    Call Algorithm 4 with $a = \gamma_{dec}$
35:    Evaluate $R^f(\mathbf{x})$ and $R^c(\mathbf{x})$ for a randomly sampled $\mathbf{x} \in B(\mathbf{x}_k, \rho_k)$.
36:  end if
37:  Update GPs and black-box evaluations.
38:  Update surrogate models $m_k^{\tilde{R}^f}(\mathbf{x}_{k+1})$, $m_k^{\tilde{R}^c}(\mathbf{x}_{k+1})$.
39: end for
```

We discuss a two-dimensional test problem that helps develop an understanding of the optimization process and the GP's role in reducing noise in Example 11. We illustrate how SNOWPAC utilizes GP surrogates and the feasibility restoration mode to identify good approximations of an optimization problem's optimal point. We also get some insight into the convergence and the feasibility-restoration mode of SNOWPAC.

Example 11 (A two-dimensional test example)**.** *We regard the test problem 228 from the CUTEst benchmark set, where we add artificial noise to create a stochastic problem and use $\mathcal{R}_0^b(x)$ as measure:*

$$\begin{aligned} \min \mathbb{E} &\left[\sin(x-1+\theta_1)+\sin\left(\frac{1}{2}y-1+\theta_1\right)^2\right]+\frac{1}{2}\left(x+\frac{1}{2}\right)^2-y, \\ s.t. \quad &\mathbb{E}\left[-4x^2(1+\theta_2)-10\theta_3\right] \qquad \leq 25-10y, \\ &\mathbb{E}\left[-2y^2(1+\theta_4)-10(\theta_4+\theta_2)\right] \leq 20x-15. \end{aligned} \tag{15.1}$$

Here, we use the random variables $\boldsymbol{\theta}=[\theta_1,\dots,\theta_4]\sim\mathcal{U}[-1,1]^4$ and set the starting point to $(x_0,y_0)=(4,3)$. To approximate the expected values, we employ $N=50$ samples of $\boldsymbol{\theta}$, and estimate the noise magnitudes using the method described in Section 14.1.

In Fig. 15.2 we plot the optimization results after 20 *(top left),* 40 *(top right), and* 100 *(bottom) optimization steps. We make multiple observations in this example: first, we observe that SNOWPAC is finding the optimal solution by first moving towards the constraint and then along the constraint. Second, we see a reduction in the trust-region radius when converging to the optimum, resulting in a more accurate surrogate. Third, we notice the effect of the noise reduction technique discussed in Section 14.3 by plotting the objective function and constraints as SNOWPAC sees them due to the correction with the GP surrogates: in the vicinity of the current design point, we clearly see smoother contour lines compared to the surrounding noisy evaluations in each of the figures. This is especially visible in the trust region. As a result, noise is significantly reduced, which facilitates efficient approximation of the optimal solution by SNOWPAC and shifts the balance towards trusting the GP estimator. The darker area indicates that more weight is given to the GP estimator, clearly showing the contribution of the GP. Moreover, we see that the optimizer eventually gathers more and more black-box evaluations, yielding an increasingly better noise reduction, also allowing a reduction in the trust region. Fourth and finally, upon analyzing the constraint contours that are affected by a significant amount of noise, it becomes apparent that relying solely on GP-supported black-box evaluations and NOWPAC's inner boundary path to ensure feasibility is not enough. Therefore, the feasibility restoration mode, as presented in Section 14.4, is necessary, which enables the optimizer to recover feasibility from seemingly infeasible points and guides the optimizer back to a feasible solution.*

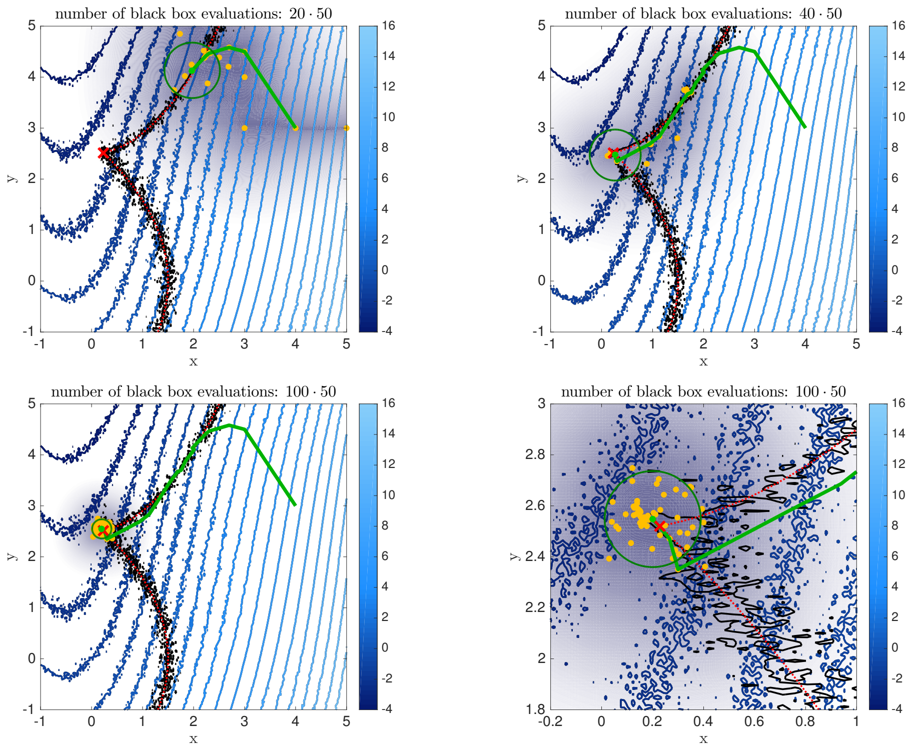


Figure 15.2: Realizations of the contour plots of the noisy objective function and constraints for the optimization problem in Eq. (15.1). The exact optimal design and constraints are indicated by a red cross and dotted red line, respecitvely. The plots show the best point (green dot) and the optimization path (green line) after 20, 40 and 100 evaluations of the robustness measures; the lower right plot is zoomed in to the neighborhood of the optimal point. The corresponding trust regions are indicated by green circles. The gray cloud indicates the size of the weighing factor γ_s^i from Eq. (14.15);the darker the area, the more weight is given to the GP. The GP regression points are indicated by yellow dots.

16 Numerical results on CUTEst benchmark

Let us now validate the performance of our method on an optimization benchmark from the CUTEst benchmark suite. We compare SNOWPAC to a variety of other methods in the field of stochastic derivative-free optimization. This includes Bayesian optimization methods, stochastic approximation methods, trust-region and direct-search methods.

Utilizing GP surrogate models in SNOWPAC connects the method with the successful class of Bayesian optimization techniques [228, 229], and their nonlinear optimization extensions that include using an augmented Lagrangian approach [146] or expected constrained improvement in the constrained Bayesian optimization (cBO) [129]. However, unlike Bayesian optimization, SNOWPAC applies GP surrogates to smooth local trust-region steps instead of concentrating on global optimization. In this section, we demonstrate how combining efficient local optimization and a second layer of GP models for smoothing produces an accurate and effective optimization technique. Furthermore, we compare the performance of SNOWPAC with optimization codes, such as COBYLA [261] and NOMAD [21], and stochastic approximation methods, such as SPSA [295, 296, 315] and KWSA [179].

Since COYBLA and NOMAD are not designed for stochastic optimization, their performance improves for smaller noise levels. We therefore vary the sample sizes to assess their performance based on different magnitudes of noise in the sample approximations of the robust objective function and constraints. We also compare the two different approaches of SNOWPAC to find the optimal γ_k^* presented in the previous Section 14.3.2, which we called the *analytic* and *heuristic* approach.

For all the following results we use a stationary squared exponential kernel

$$k^b(\mathbf{x}, \mathbf{x}') = (\sigma_f^b)^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^T \mathbf{\Lambda}_{l^b}^{-1}(\mathbf{x}-\mathbf{x}')\right), \tag{16.1}$$

for the construction of the GP surrogates in the upcoming sections with $(\sigma_f^b)^2 \in \mathbb{R}_+$ and $\mathbf{l}^b = \text{diag}(\mathbf{\Lambda}_{l^b})$ with $\mathbf{l}^b = [l_1^b, ..., l_{d_{\text{det}}}^b]^T \in \mathbb{R}_+^{d_{\text{det}}}$ as the hyperparameters of the GP. To determine the hyperparameters, we maximize the marginal likelihood of the estimator using automatic relevance determination (ARD) [269, 318]. This means, we use different length-scale hyperparameters for each dimension $l_i^b, i = 1, ..., d_{\text{det}}$. We estimate the hyperparameters after a fixed number of optimization steps.

16.1 Stochastic problem setting

To evaluate the performance of all optimizers, we utilize the Schittkowski optimization benchmark set [158, 282], which is a part of the CUTEst benchmark suite for non-linear constraint optimization. We pick the following eight problems: $TP \in \mathcal{TP} = \{29, 43, 100, 113, 227, 228, 268, 285\}$. This test set includes feasible domains with dimensions ranging from 2 to 16 and a variable number of constraints ranging from 1 to 10. The deterministic selection of problems is given in the Appendix D.4.

As the problems are deterministic, we introduce noise to the objective functions $(f(\mathbf{x}) + \theta_0)$ and constraints $(\{c_i(\mathbf{x}) + \theta_i\}_{i=1}^r)$ where $(\theta_0, \theta_1, ..., \theta_r) \sim \mathcal{U}[-1, 1]^{1+r}$, and we solve three classes of robust optimization problems as follows:

1. Minimization of the average objective function subject to the constraints being satisfied in expectation:
$$\begin{aligned} \min \mathcal{R}_0^f(\mathbf{x}) \\ \text{s.t.} \quad \mathcal{R}_0^c(\mathbf{x}) \leq 0. \end{aligned} \tag{16.2}$$

2. Minimization of the average objective function subject to the constraints being satisfied in 95% of all cases:
$$\begin{aligned} \min \mathcal{R}_0^f(\mathbf{x}) \\ \text{s.t.} \quad \mathcal{R}_4^{c,0.95}(\mathbf{x}) \leq 0. \end{aligned} \tag{16.3}$$

3. Minimization of the 95%-CVaR of the objective function subject to the constraints being satisfied on average:
$$\begin{aligned} \min \mathcal{R}_5^{f,0.95}(\mathbf{x}) \\ \text{s.t.} \quad \mathcal{R}_0^c(\mathbf{x}) \leq 0. \end{aligned} \tag{16.4}$$

16.2 Data profile metric

The authors [232] introduce *data profiles* to quantify the performance of the different solvers over a collection of optimization runs, which we will use in this work as well. To create this collection for the performance comparison, we solve each of the eight optimization problems in $\mathcal{TP}$ with 100 random initial points and three different MC sample sizes $N \in \{200, 1000, 2000\}$. Thus, we get a total number of $3 \cdot 8 \cdot 100 = 2400$ optimization runs. We then denote the resulting benchmark set the collection of all results $\mathcal{P}$. The accuracy requirement for the objective and constraints function is given by the tolerances ϵ_f and ϵ_c and defined as

$$\frac{\left|\mathcal{R}^f(\mathbf{x}_k) - \mathcal{R}^f(\mathbf{x}^*)\right|}{\max\{1, |\mathcal{R}^f(\mathbf{x}^*)|\}} \leq \epsilon_f \quad \text{and} \quad \max_{i=1}^{r} \left\{[\mathcal{R}^{c_i}(\mathbf{x}_k)]^+\right\} \leq \epsilon_c. \tag{16.5}$$

To generate the data profiles, we calculate the minimum number of optimization steps, denoted as $t_{p,S}$, required by solver S to meet this accuracy requirement for solving problem $p \in \mathcal{P}$. We restrict the maximum number of optimization steps to 250 and assign $t_{p,S} = \infty$ if the accuracy requirement is not satisfied after $250 \times N$ black box evaluations. To determine if the accuracy requirement is met, we employ the precise values of the robustness measures obtained in a post-processing phase. Finally, this results in the data profile

$$d_S(\alpha) = \frac{1}{2400} \left| \left\{ p \in \mathcal{P} \ : \ \frac{t_{p,S}}{n_p + 1} \leq \alpha \right\} \right|, \tag{16.6}$$

where n_p denotes the number of design parameters in problem p. We would like to point out that while using the exact objective and constraint values of the robustness measures in the post-processing step helps to eliminate the influence of noise on performance evaluation, this deterministic information is generally not available. Therefore, we are also going to provide a more detailed analysis of individual optimization results below in the second part of this results section.

16.3 Data profile results

For all resulting figures in section, we use the same visualization. For SNOWPAC, we show two approaches, showcasing the choice for the GP smoothing: analytic ("altyc", pink, dashed dot) and heuristic ("heur", red, dotted). We compare to cBO (blue, solid), COBYLA (purple, dashed), NOMAD (green, dotted), SPSA (orange, solid), and KWSA (dark green, dashed dash). We start with the data profiles for the error thresholds $\epsilon_f \in \{1\text{e-}2, 1\text{e-}3\}$ and $\epsilon_c \in \{1\text{e-}2, 1\text{e-}3\}$ in Figs. 16.1-16.3 for the three problems from Eq. (16.2), Eq. (16.3) and Eq. (16.4).

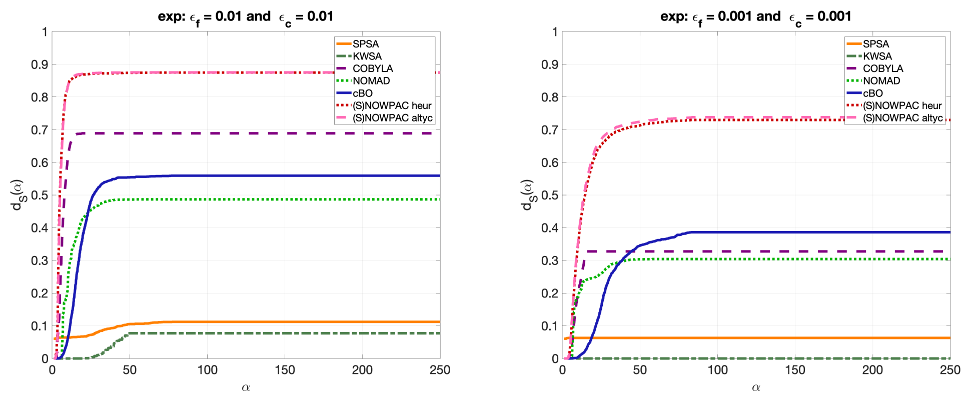


Figure 16.1: Data profiles for Eq. (16.2) with the given tolerances $\epsilon_{f/c} \in \{1\text{e-}2, 1\text{e-}3\}$.

We observe that both SNOWPAC methods effectively solve the majority of test problems within the designated budget of black box evaluations. For a tolerance of $\epsilon_f = 1\text{e-}2$ and $\epsilon_c = 1\text{e-}2$, we see that we solve more than 85% of the problems in less than 50 (relative[1]) steps. SNOWPAC is also able to solve the most problems compared to the other methods, which is indicated by the highest value of the data profile over all tolerances and robust formulations. When examining the performance for low values of α, we notice that SNOWPAC demonstrates comparable or better performance compared to the other solvers: SNOWPAC shows similar or even the steepest and also earliest ascent in the profile. This indicates a fast search for the local optima, which is especially desirable when computing the robustness measures is computationally costly. The performance of all methods worsens with decreasing tolerance, which is reasonable due to the higher requirement in accuracy. Nevertheless, also for smaller tolerances, SNOWPAC still performs best.

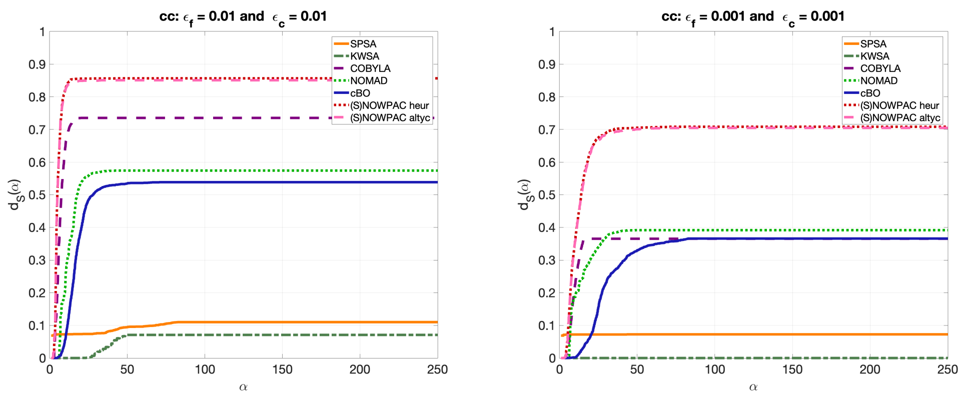


Figure 16.2: Data profiles for Eq. (16.3) with the given tolerances $\epsilon_{f/c} \in \{1\text{e-}2, 1\text{e-}3\}$.

COBYLA and NOMAD perform well for larger thresholds that are of the same magnitudes as the noise term in some test problems. Especially the results of COBYLA decrease notably for tolerances ϵ_f = 1e-3 and ϵ_c = 1e-3. The noise reduction in SNOWPAC using the GP support helps to approximate the optimal solution more accurately, resulting in better performance results. The stochastic approximation approaches SPSA and KWSA, despite a careful choice of hyperparameters, do not perform well on the benchmark problems. This can be explained by the limited number of overall optimization iterations, which are not sufficient to achieve a good approximation of the optimal solution using inaccurate gradients.

We cannot see, however, a big difference between the two SNOWPAC approaches. It is also challenging to see where the individual algorithms perform best and if there are certain problems where they perform worse. To further investigate this, we decrease the tolerance even further to ϵ_f = 1e-4 and ϵ_c = 1e-4 and plot the result for the data profiles in Fig. 16.4. While the performance of all methods is worse for such a low tolerance, we now see a visible improvement when using the analytic approximation in SNOWPAC for Eq. (16.2) (top left) and Eq. (16.4) (bottom). At this high level of accuracy, the analytic approximation is the most reliable approach to find the optimal solution.

[1]Relative, since the number of steps is scaled by the dimension of the problem.

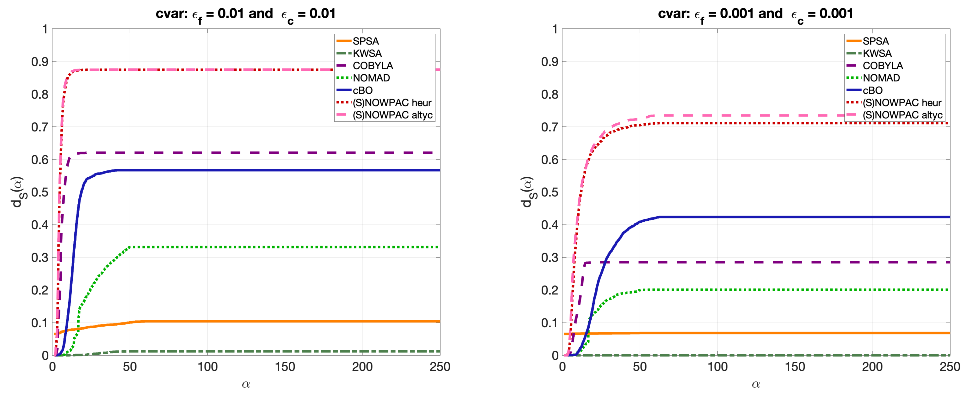


Figure 16.3: Data profiles for Eq. (16.4) with the given tolerances $\epsilon_{f/c} \in \{1\text{e-}2, 1\text{e-}3\}$.

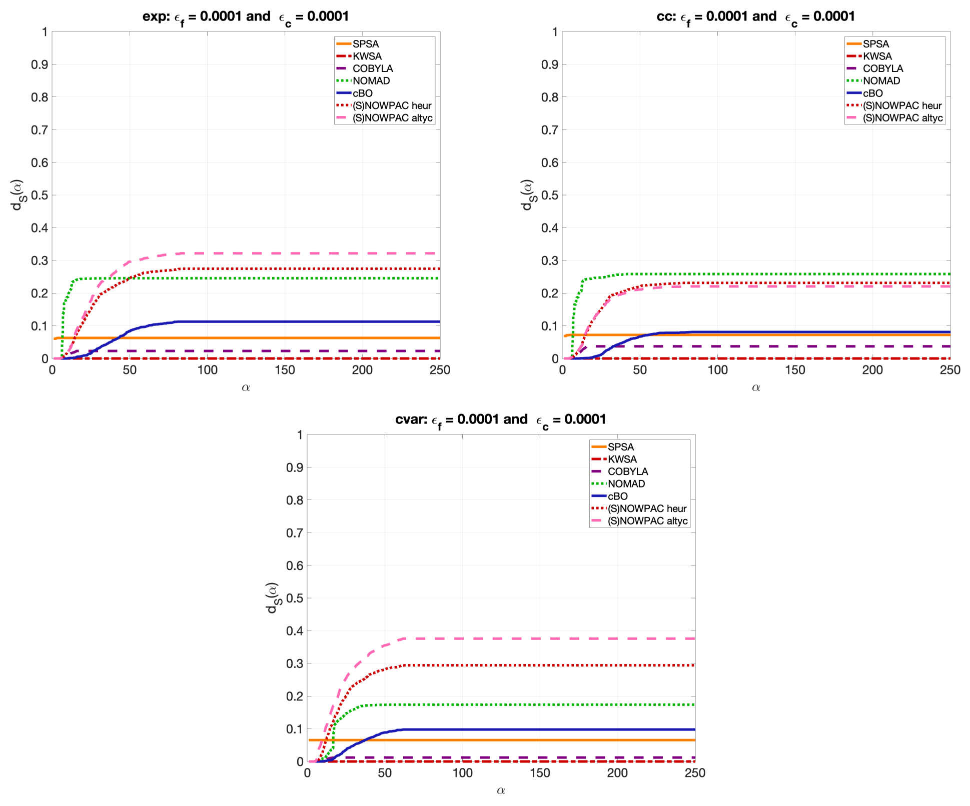


Figure 16.4: Data profiles for Eqs. (16.2) (top left), (16.3) (top right) and (16.4) (bottom) and thresholds $\epsilon_f = \epsilon_c = 1\text{e-}4$.

16.4 Results for individual benchmark problems

For further insight, we show a detailed accuracy comparison of the individual optimization results at termination, i.e. $250 \cdot N$ black box evaluations in Figs. 16.5 and 16.6. Here, we show the accuracy of the optimization results at the approximated optimal points at termination of the optimizers. The plots show the errors in the objective values, the constraint violations and the errors in the approximated optimal designs found by the optimizers at termination respectively. Since the optimal solution for test problem 268 is zero, we show the absolute error for this test problem. We use MATLAB's box plots to summarize the results for 100 optimization runs for each benchmark problem for different sample sizes $N \in \{2000, 1000, 200\}$ from left to right separately for each individual robust formulation Eq. (16.2)-(16.4). The exact evaluation of the robust objective function and constraints at the approximated optimal designs are shown to eliminate the randomness in the qualitative accuracy of the optimization results. We denote the approaches for SNOWPAC as *(S) a* for the analytic approximation and *(S) h* for the heuristic due to space limitations.

We see that SNOWPAC most reliably finds accurate approximations to the exact optimal solutions. This is indicated by the lowest error for almost all test problems in the objective (left column) as well as in the final design (right column). Note that all optimizers benefit from increasing the number of samples for the approximation of the robustness measures. In SNOWPAC, however, the GP surrogates additionally exploit information from neighboring points to further reduce the noise, allowing for a better accuracy in the optimization results. We expected SNOWPAC to perform worse for higher dimensions due the necessity for more evaluations for high accuracy in the surrogate as well as in the GP. While we see worse results for problem 100 (seven dimensions) and 285 (15 dimensions), where COBYLA performs best, we see good results for problem 113 (ten dimensional), where SNOWPAC performs best. Hence, the accuracy is most likely connected to the GP and to the choice of kernel.

We confirm this by regarding the result for cBO, which fully depends on GPs and does not use an additional surrogate. The results match well between SNOWPAC and cBO for low-dimensional problems 29, 227, 228, but the accuracy of the results computed by cBO begins to deteriorate in dimensions larger than 4. For cBO this has two reasons: first, the global search strategy aims at variance reduction within the whole search domain. This requires more function evaluations than in a local search, which we test here. Second, the global nature of the GPs requires a suitable choice of kernels that fits to the properties of the optimization problems, i.e. non-stationarity of the optimization problem, which is not the case in all benchmark problems. A stationary kernel as given in Eq. (16.1), may not properly reflect the properties of the objective functions and constraints. Since SNOWPAC uses GPs only for correction and combines it with a local surrogate, SNOWPAC reduces the problem of violated stationarity assumptions on the objective function and constraints, the downside of GPs is mitigated and the method still is able to perform well for higher dimensions while similar challenges like kernel choice are still evident. A problem dependent choice of kernel function might help to reduce this problem, however, this information is often hard to obtain in black-box optimization.

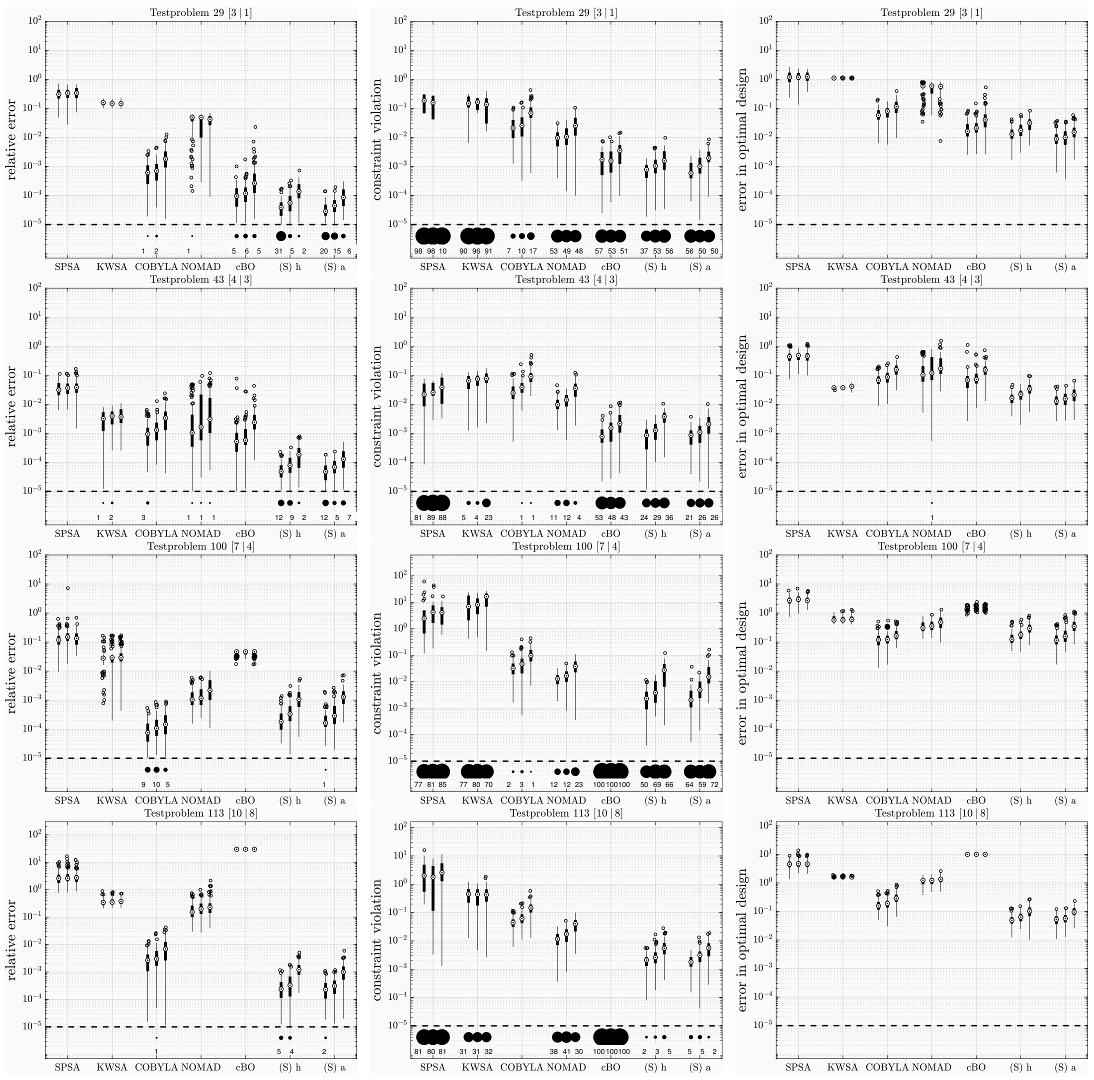


Figure 16.5: Box plots of the errors in the approximated optimal objective values (left plots), the constraint violations (middle plots) and the l_2 distance to the exact optimal solution (right plots) for the test problems 29, 43, 100, and 113 for Eq.(16.2). The plots compare results for SNOWPAC analytic ((S) a), SNOWPAC heuristic ((S) h), cBO, COBYLA, NOMAD, SPSA and KWSA. All errors or constraint violations below 1e-5 are stated separately below the threshold; the box plots only contain data above this threshold.

We are going to regard this in an extension of SNOWPAC for adaptive kernel selection in Section 17.1.

The middle column in Fig. 16.5 and 16.6 show the maximal constraint violations at the approximated optimal designs. Here, SNOWPAC's constraint handling, see [23], in

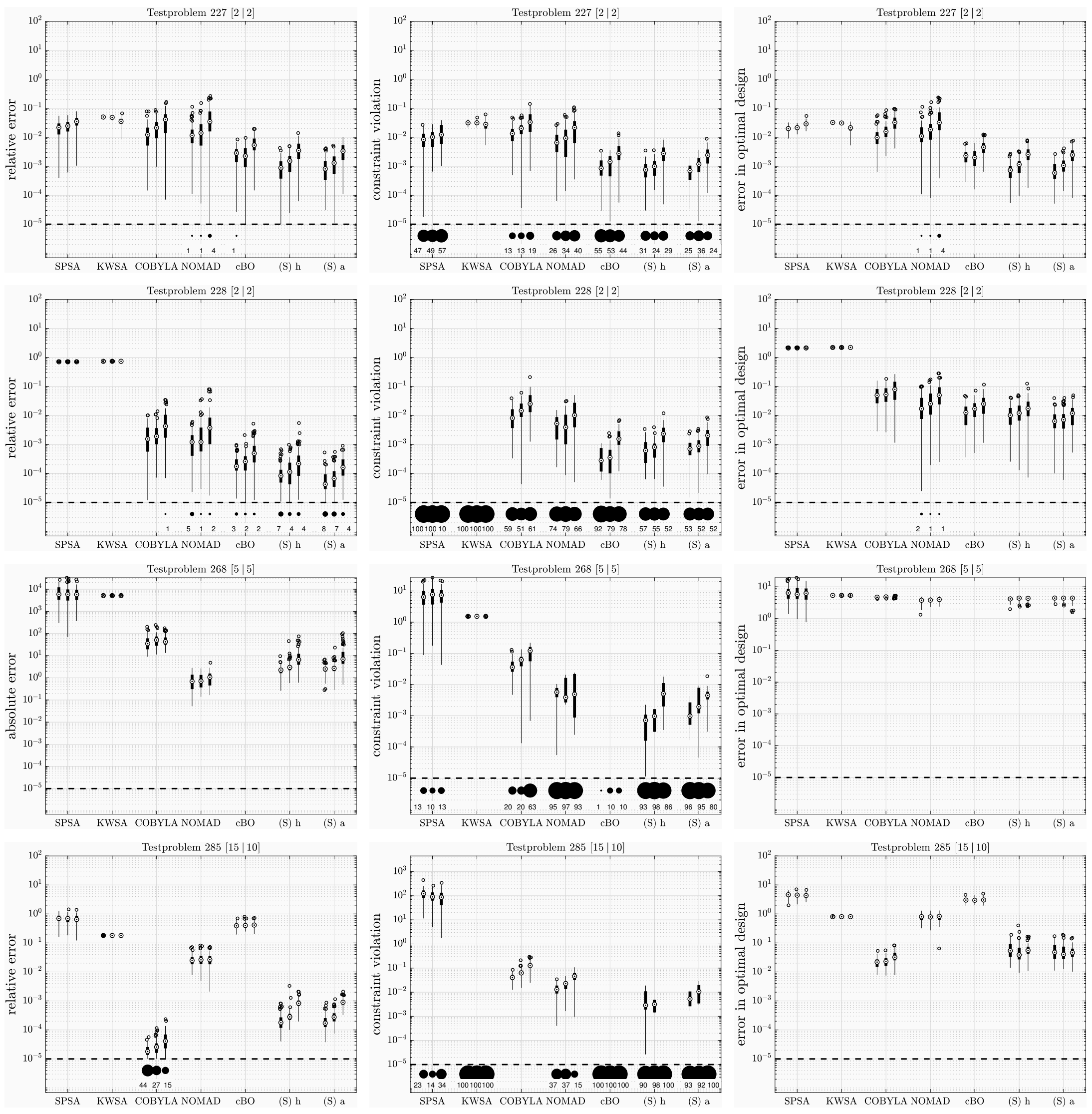


Figure 16.6: Box plots of the errors in the approximated optimal objective values (left plots), the constraint violations (middle plots) and the l_2 distance to the exact optimal solution (right plots) for the test problems 227, 228, 268, and 285 for Eq. (16.2). The plots compare results for SNOWPAC, cBO, COBYLA, NOMAD, SPSA and KWSA. All errors or constraint violations below 1e-5 are stated separately below the threshold; the box plots only contain data above this threshold.

combination with the feasibility-restoration mode from Section 14.4 allows the computation of approximate optimal designs that exhibit only small constraint violations well below the noise level. SNOWPAC, on the one hand, tries to find a balance between the constraint violation acceptable due to noise, while being as close to the optimum as possible. The methods SPSA, KWSA and also cBO, on the other hand, show very little constraint violations which, however, results in suboptimal solutions found for the objective as well as the design. This is indicated by the black dots and numbers below the constraint violation of 1e-5. Given the stochastic nature of the problem, we allow SNOWPAC minor inaccuracy in the constraints to allow for a better optimum (and still control large deviations with the feasibility-restoration mode).

Finally, comparing the analytic and heuristic approach, we see that the analytic approach shows a lower relative error in the objective and the optimal design, especially for lower dimensional problems, which was also reflected in the data profile for ϵ_f = 1e-4 and ϵ_c = 1e-4 in Fig. 16.4. This is due to the fact that the GP surrogates work especially well in lower dimensions and therefore the optimal smoothing parameter is well approximated. The improvement is, e.g., visible in test problem 29 and 228 of Fig. 16.5 and Fig. 16.6. The analytic smoothing, nevertheless, also shows similar or even better results for high-dimensional problems 100, 113 and 285. Combined with the results mentioned above, this shows the validity of both approaches. We can decide which one to use, based on the problem and the available computational resources.

16.5 Summary and challenges of SNOWPAC

This closes the first part of our second contribution, SNOWPAC. We showed the applicability of the method for derivative-free stochastic black-box problems. This required an extension to the deterministic method, NOWPAC, where we adapt the method to be able to adjust for noise evaluations, employing an adapted trust-region management, a GP surrogate for smoothing and a feasibility restoration mode. We also presented that the method is able to compete with other approaches in the field, showing best results for different problem formulations, sample sizes, deterministic and stochastic dimensions.

Still, challenges remain for the method. One challenge is the choice of kernel, which impacts the overall performance of the surrogate and is connected to the problem. Another challenge is the rising computational cost with the available number of evaluations. The evaluation cost of the GPs increases cubically with the number of evaluation point due to the inversion of the kernel matrix. Also, the hyperparameter optimization of the GP in SNOWPAC scales with the optimization steps. Finally, we only handled continuous problems so far, but discrete problems are also of great interest in the field of optimization, especially with rising interest in applications in machine learning. We will discuss extensions to SNOWPAC to be able to handle such challenges in the next chapter.

17 Extensions for SNOWPAC

This chapter encompasses extensions to SNOWPAC to be able to cope with the previously mentioned challenges. In the upcoming sections, we will present implementations of approximate GP methods in SNOWPAC to mitigate the decrease in performance for higher number of iterations in Section 17.1. Subsequently, we will present an adaptive choice of kernel for the GP surrogates in SNOWPAC in Section 17.2. Lastly, SNOWPAC was developed for continuous problems. Nevertheless, we often encounter problems which also require optimization over discrete variables. Here, a common and fast growing field is neural network architecture search in machine learning, where we optimize for the optimal design of a neural network. The parameters of the neural network itself are often continuous, like the learning rate or the connection weights, but the parameters of the neural network architecture comprise the number of hidden layers or the number of neurons—discrete parameters. Hence, we will present an extension of SNOWPAC for mixed-integer problems in the final Section 17.3.

17.1 Adaptive Gaussian process kernel selection in SNOWPAC

The choice of kernel plays an important role in the approximation quality of a GP. By choosing the kernel, we set a prior for the function space that we want to approximate, where kernels exist for (non-)smooth functions, for periodic functions, (in-)stationary or even discontinuous functions. (See [103] for a broad selection of kernels.) Also, combinations of kernels were explored in [102, 174, 220]. We discussed properties of kernels and gave different examples in Section 7.3.

In this section, we investigate if and how the choice of certain kernels impacts the performance in SNOWPAC. Since the GP plays an important role in the approximation quality of the surrogate, we expect to require different kernels for the best approximation. In this section, we even move a step further: instead of just testing different kernels, we explore a strategy of automatically adapting the kernel based on the current best fit. This work is based on a first explorative study for different kernels by the bachelor student Marina Baumgartner starting from kernels presented in [269].

Kernel selection

In this study, we implemented four different kernel, where we define the squared distance between points $\mathbf{x}$ and $\mathbf{x}'$ as $r(\mathbf{x},\mathbf{x}') = \sum_{i=1}^{d_{\text{det}}} \frac{(x_i - x_i')^2}{l_i^2}$ scaled by the characteristic length scale $\{l_i\}_{i=1}^{d_{\text{det}}}$:

- Squared exponential:
$$k_{\text{SE}}(\mathbf{x},\mathbf{x}') = \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^T \mathbf{\Lambda}_l^{-1}(\mathbf{x}-\mathbf{x}')\right). \tag{17.1}$$
This is the default kernel of SNOWPAC we have already presented in Eq. (16.1) and repeat here for convenience[1]. The squared exponential kernel is the standard choice in GPs. It is a smooth kernel, infinitely differentiable, with $d_{\text{det}}+1$ hyperparameters for the variance σ and the d_{det} length-scale parameters l_i.

- Matérn:
$$k_{\text{Matern},\nu}(\mathbf{x},\mathbf{x}') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu r(\mathbf{x},\mathbf{x}')}\right)^{\nu} K_\nu \left(\sqrt{2\nu r(\mathbf{x},\mathbf{x}')}\right). \tag{17.2}$$
Here, K_ν is the modified Bessel function [1] and $\Gamma(\nu) = \int_0^\infty t^{\nu-1}e^{-t}\text{d}t$ is the Gamma function. The Matérn kernel is a widely-used choice because of its adaptability due to ν. It is k-times mean square differentiable if and only if $\nu > k$. We have $d_{\text{det}}+2$ hyperparameters with σ, $\{l_i\}_{i=1}^{d_{\text{det}}}$ and ν.

  There are a few special cases: the Matérn kernel represents a rather non-smooth function for the choice $\nu = \frac{3}{2}$, which results in
$$k_{\text{Matern},\nu=\frac{3}{2}}(\mathbf{x},\mathbf{x}') = \left(1+\sqrt{3}r(\mathbf{x},\mathbf{x}')\right)\exp\left(-\sqrt{3}r(\mathbf{x},\mathbf{x}')\right), \nu > 0. \tag{17.3}$$
The parameter $\nu = \frac{5}{2}$ was used to model discontinuities in [84] and can be computed as
$$k_{\text{Matern},\nu=\frac{5}{2}}(\mathbf{x},\mathbf{x}') = \left(1+\sqrt{5}r(\mathbf{x},\mathbf{x}')+\frac{5}{3}r(\mathbf{x},\mathbf{x}')^2\right)\exp\left(-\sqrt{5}r(\mathbf{x},\mathbf{x}')\right). \tag{17.4}$$
There is another special case for $\nu = \frac{1}{2}$, which results in the exponential covariance kernel
$$k_{\text{Matern},\nu=\frac{1}{2}}(\mathbf{x},\mathbf{x}') = \exp\left(r(\mathbf{x},\mathbf{x}')\right). \tag{17.5}$$
For $d_{\text{det}} = 1$, this is the covariance function of the Ornstein-Uhlenbeck process, which models the velocity of particles in Brownian motion [308].

- Rational quadratic:
$$k_{\text{RQ}}(\mathbf{x},\mathbf{x}') = \sigma^2\left(1+\frac{r(\mathbf{x},\mathbf{x}')}{2\alpha}\right)^{-\alpha}, \alpha > 0. \tag{17.6}$$
This kernel has $d_{\text{det}}+2$ hyperparameters with σ, $\{l_i\}_{i=1}^{d_{\text{det}}}$ and α. The properties of the kernel change by changing α. Knowing that $\exp(x) = \lim_{d_{\text{det}}\to\infty}\left(1+\frac{x}{d_{\text{det}}}\right)^{d_{\text{det}}}$, we deduce that the rational quadratic kernel converges to the squared exponential kernel (17.1) for $\alpha \to \infty$. For smaller values of α, the kernel function is able to represent less smooth functions.

[1]without the notation b because we do not need to differentiate between objective and constraints.

- Exponential sine squared:

$$k_{\text{ESS}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\sum_{i=1}^{d_{\text{det}}} \frac{\sin^2(|\mathbf{x}_i - \mathbf{x}'_i| \frac{\pi}{p})}{l_i}\right), p > 0. \tag{17.7}$$

  Due to the use of the sine, the kernel is able to reproduce periodic functions [103, 215]. The period p determines the distance between repetitions of the functions, while we again have d_{det} parameters for the length scale and one parameter for the noise as σ.

Kernel optimization in SNOWPAC

As stated in the previous section, SNOWPAC employs maximization of the marginal log-likelihood to optimize the hyperparameters of the kernel function. To find the optimal kernel, we run the hyperparameter optimization for each kernel. We pick the kernel that achieves the largest marginal log-likelihood, therefore adapting best for the given data, with the given hyperparameters. As a result, we employ different kernels for different sections of the optimization and different kernels for the different functions, i.e. objective and constraints. Formally, we write this problem as

$$\min_{k \in \{k_{\text{SE}}, k_{\text{Matern},\nu}, k_{\text{RQ}}, k_{\text{ESS}}\}} \frac{1}{2} \log |\mathbf{K}_{\mathbf{XX}} + \sigma_n^2 \mathbf{I}| + \frac{1}{2} \mathbf{y}^T (\mathbf{K}_{\mathbf{XX}} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \tag{17.8}$$

where the kernel matrix $\mathbf{K}_{\mathbf{XX}}$ is build using k on each pair of data points.

The timing of the hyperparameter optimization is set by the user and can either be run periodically by setting `GP_adaption_factor` or at specific optimization steps by setting `GP_adaption_steps`. The functionality of selecting the adaptive kernel choice is available in SNOWPAC by setting the option `Kernel_type` to `Kernel_Opt`. The GP option `GP_Type` has to be set to `GP` for this setting. Apart from the adaptive setting, specific kernels can also be set for the full optimization by setting `Kernel_type` to `SE` (default), `Matern`, `ESS` or `RQ`.

Benchmark results

Let us have a look at the CUTEst benchmark results from before in Chap. 16, now also comparing the kernel optimization. To downsample the results, we restrict ourselves to the results for optimizing for $\mathcal{R}_0^f$ as given in (16.2). The kernel optimization is triggered every 50 iterations. We show the data profiles for two tolerances $\epsilon_f = \epsilon_c =$ 1e-2 and $\epsilon_f = \epsilon_c =$ 5e-4. We added a new line for SNOWPAC using the kernel optimization as *SNOWPAC KO* (pink, dashed). We compare to heuristic SNOWPAC (red, dotted), cBO (blue, solid), COBYLA (purple, dashed), NOMAD (green, dotted), SPSA (orange, solid), and KWSA (dark green, dashed dash). The figure is given in Fig. 17.1.

We first observe that both approaches for SNOWPAC perform similarly for the lower tolerance $\epsilon_f = \epsilon_c =$ 1e-2. Similar to the comparison with the analytic solution, we, however, see a difference when decreasing the tolerance to $\epsilon_f = \epsilon_c =$ 5e-4. To reach

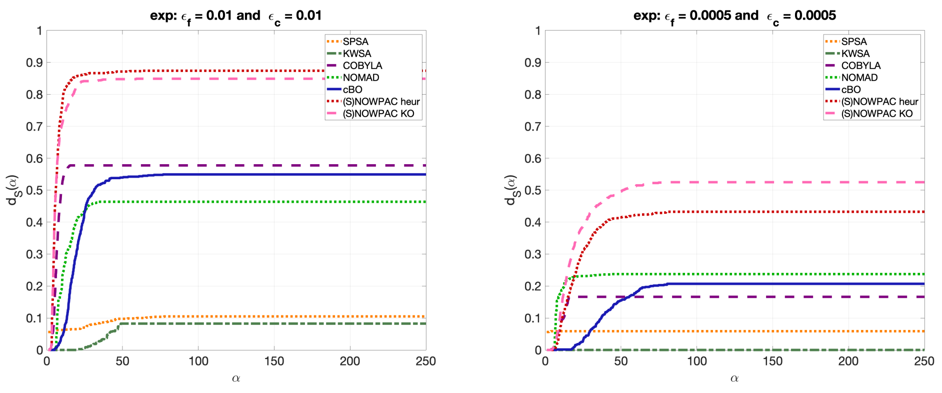


Figure 17.1: The plots compare results for SNOWPAC using kernel optimization ((S) ko), SNOWPAC heuristic ((S) h), cBO, COBYLA, NOMAD, SPSA and KWSA. The data profiles for thresholds $\epsilon_f = \epsilon_c =$ 1e-2 and $\epsilon_f = \epsilon_c =$ 5e-4 on the objective values and the constraint violation are shown on the left and right, respectively.

this low tolerance, the SNOWPAC needs more optimization iterations.This is directly connected to the accuracy of the GP, which improves with the number of iterations, i.e. the number of training points. For a more accurate GP, the kernel and hyperparameter optimization is more accurate as well, which in turn results in improved results.

We now investigate further and look at the performance for individual problems. Specifically, we regard the higher dimensional problems 100, which has seven design dimensions and four constraints, and 285, which has 15 design dimensions and ten constraints. The results are given for the absolute error in the objective, the constraint violation and the error in the design in the boxplots in Fig. 17.2. We use the abbreviation *(S) ko* for the kernel optimization in SNOWPAC. The plots compare results for SNOWPAC using kernel optimization ((S) ko), SNOWPAC heuristic ((S) h), cBO, COBYLA, NOMAD, SPSA and KWSA.

We note that using the kernel optimization improves the results in the objective and in the design for both problems. For problem 285, we see a minor increase in the constraint violation. Nonetheless, it seems that the kernel optimization shows promises for higher dimensional problems, which might be challenging for the standard GP using the squared exponential kernel only. Another point is that we allow more optimization steps for higher dimensional problems resulting in a better approximation of the GP.

As a last observation for these results, we regard which kernels are actually chosen for which function based on the optimization of the marginal likelihood. For this, we again look at the two test problems 100 and 285. We take the runs we have used for plotting the data profiles where we have a set of 300 optimization runs for each test problem (100 sample runs for three different sample sizes). For each run, we evaluate 250 optimization steps and average the kernel used in the specific step over the 300 runs by assigning a value of one to four to the kernel ($1 = K_{\mathrm{SE}}$, $2 = K_{\mathrm{Matern}}$, $3 = K_{\mathrm{ESS}}$, $4 = K_{\mathrm{RQ}}$). We plot

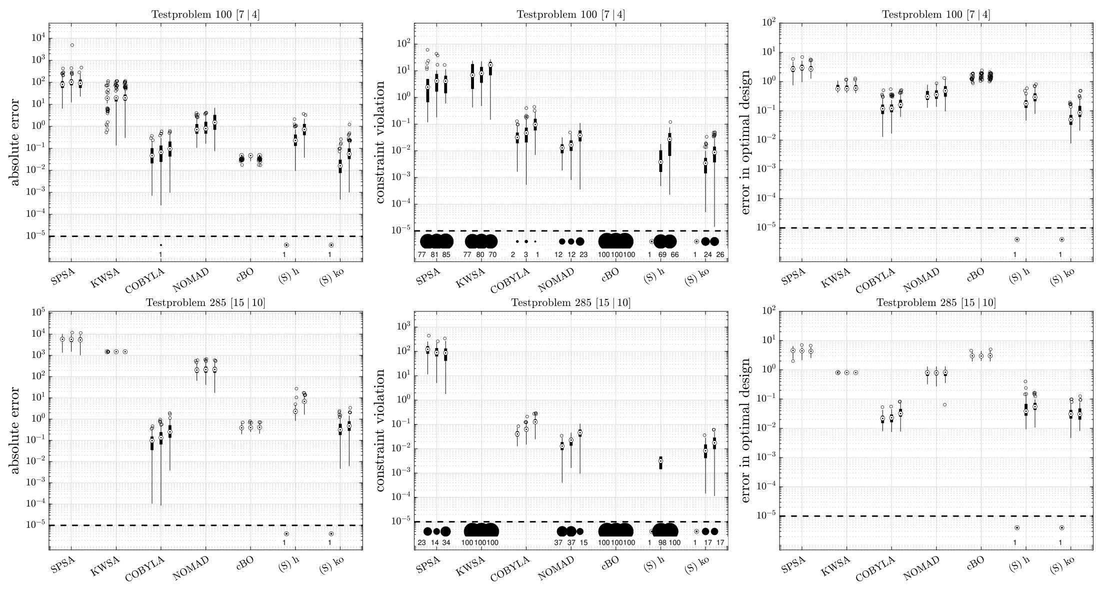


Figure 17.2: Box plots of the errors in the approximated optimal objective values (left plots), the constraint violations (middle plots) and the l_2 distance to the exact optimal solution (right plots) for the test problems 100 and 285 for (16.2). All errors or constraint violations below 1e-5 are stated separately below the threshold and the box plots only contain data above this threshold.

the resulting averaged kernels over the 250 optimization steps for each of the functions in the test problem—objective and constraints. We illustrate the results in Fig. 17.3.

We see a change in the kernel almost every 50 optimization steps, i.e. after almost every hyperparameter optimization. Notably, the different objective and constraint functions indeed use different kernels and change kernels during the optimization, which is an indicator that the choice of kernel is important for the function fit. Most of the functions use some variation of the Matérn kernel or the ESS kernel, whereas the RQ kernel is not used for those two problems.

We summarize the usage of different kernels over the full benchmark set in Fig. 17.4. Here, we show the percentage of kernels used over all problems and optimization runs. The Matérn kernel is the most prevalent kernel, followed by the SE and the ESS kernel. In our test set, we only had a small amount of specific runs where the rational quadratic kernel was used. The Matérn kernel offers more flexibility and the ESS kernel covers periodic functions.

The result show that the option for multiple kernels in SNOWPAC provides great flexibility for the GP surrogates in SNOWPAC to adapt to the specific problem. The adaptive optimization for kernels also shows promising results in this set of benchmark problems, specifically for higher dimensional problems. This, combined with the possibility to set the steps for the hyperparameter optimization adaptively at specified

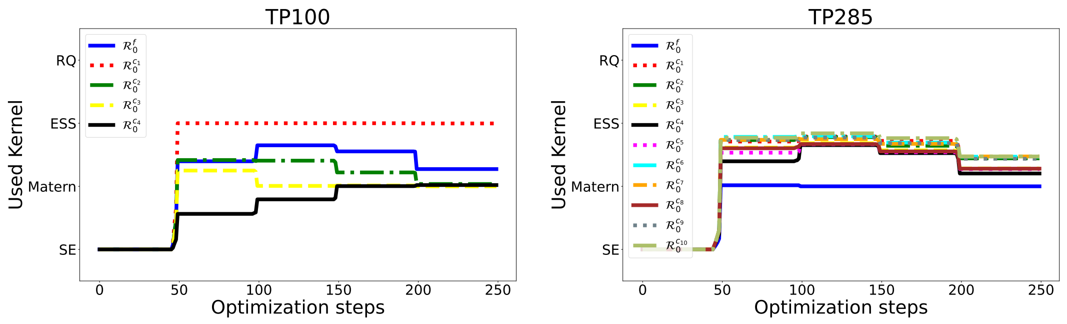


Figure 17.3: Evolution of the used kernel for test problem 100 (left) and 285 (right) for each function—objective and constraints. We show the optimization steps on the x-axis and the kernel used on the y-axis.

iterations, allows SNOWPAC to adapt better to the underlying problem and improve optimization results.

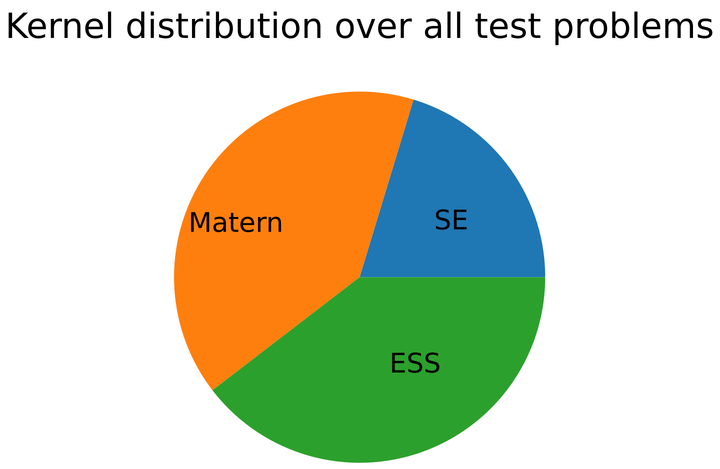


Figure 17.4: Distribution of all kernel over the full benchmark set in the bottom.

17.2 Approximate Gaussian processes in SNOWPAC

GPs are a main component in SNOWPAC for the surrogate to be able to perform well despite noisy evaluations. The evaluations of the mean, $\mathcal{G}_k^b$, and variance, $\mathcal{V}_k^b$, as denoted in Eq. 14.12, possess a specific characteristic: the kernel matrix $(\mathbf{K_{XX}} + \sigma_n^2\mathbf{I})$ must be inverted during the calculation. The size of this kernel matrix expands in line with the number of training points, causing the computation of its conventional inverse to scale cubically, i.e. in $\mathcal{O}(N^3)$, where N presents the total number of these training points. Furthermore, the inverse computation may trigger numerical errors, potentially leading to the kernel matrix losing its positive semi-definite property.

The same computational restrictions hold for the hyperparameter optimization. The most typical strategy is to estimate the hyperparameters by optimizing over the marginal likelihood as outlined in Section 7.4. Once more, the optimization process requires computing the inverse and determinant in each evaluation step. (However, it is worth noting that, from an implementation perspective, the determinant can be obtained free of charge by utilizing the Cholesky factorization from the inversion.)

Consequently, with interest in GP methods increasing, naturally, interest in improving the performance of the method by reducing computational cost also rose. In response to the costly inversion, the development of *approximate Gaussian process* (aGPs) methods has been an innovative solution [268, 269]. In the sections to follow, we will delve into aGPs, provide an introduction to hyperparameter estimation within these methods and present the usage of aGPs in SNOWPAC based on the work in [222].

Reduced-rank approximation

The most successful methods in terms of both accuracy and performance, when compared to the full GP, leverage a low-rank approximation of the matrix $\mathbf{K}$. For instance, if the kernel matrix $\mathbf{K}$ has a rank Q, it can be expressed as $\mathbf{K} = \mathbf{Q}\mathbf{Q}^T, \mathbf{Q} \in \mathbb{R}^{N\times Q}$. This representation can accelerate the matrix inversion computation by applying the Woodbury formula (refer to App. B.1.1).

$$(\mathbf{Q}\mathbf{Q}^T + \sigma_{\mathbf{n}}{}^2\mathbf{I}_n)^{-1} = \sigma_{\mathbf{n}}{}^{-2}\mathbf{I}_n - \sigma_{\mathbf{n}}{}^{-2}\mathbf{Q}(\sigma_{\mathbf{n}}{}^2\mathbf{I}_n + \mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T. \tag{17.9}$$

This approach is particularly useful for a degenerate kernel. Nevertheless, it can still prove accurate even for a nondegenerate kernel, which might exhibit a rapidly decaying eigenspectrum, characterized by swiftly diminishing eigenvalues, where such a low-rank approximation remains effective.

The optimal reduced-rank approximation of $\mathbf{K}$, in terms of the Frobenius norm, is denoted by $\mathbf{K} = \mathbf{S}\boldsymbol{\Gamma}\mathbf{S}^T$. Here, $\boldsymbol{\Gamma} \in \mathbb{R}^{M\times M}$ is the diagonal matrix of the principal M eigenvalues, and $\mathbf{S} \in \mathbb{R}^{N\times M}$ represents the matrix of the corresponding orthonormal eigenvectors [141]. However, such an eigendecomposition is also computationally demanding, involving $\mathcal{O}(N^3)$ operations.

An alternative, more commonly utilized approach, which instigated the development of approximation methods, relies on a subset of the data, denoted as $\mathbf{U} \in \mathbb{R}^{M\times d_{\text{det}}}$. It employs the *Nyström* construction as follows:

$$\mathbf{K}_{\mathbf{XX}} \approx \mathbf{K}_{\mathbf{XU}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K}_{\mathbf{UX}}, \tag{17.10}$$

where $\mathbf{K}_{\mathbf{XU}} \in \mathbb{R}^{N\times M}$ represents the kernel function evaluated on the training data and the subset $\mathbf{U}$. This subset is often also referred to as the *active set* or *induced points*. Detailed derivations of the method can be found in [324], with the method and its properties further explored in [323]. Since we will consistently be employing this matrix decomposition approximation in the upcoming sections, we introduce the notation $\mathbf{K_{AB}} \approx s\mathbf{Q_{AB}} = \mathbf{K_{AU}}\mathbf{K_{UU}^{-1}}\mathbf{K_{UB}}$, where the set of induced points are defined as $\mathbf{U} \in \mathbb{R}^{M \times d_{\text{det}}}$.

Finite-dimensional approximation

A distinct perspective is presented by [292], focusing on a specific point $\mathbf{x}_i$. The aim is to approximate the kernel as a linear combination of all kernels from the active set:

$$k(\mathbf{x}_i, \mathbf{x}) \approx \sum_{j \in\ I} c_{ij} k(\mathbf{x}_j, \mathbf{x}) =: \hat{k}(\mathbf{x}_i, \mathbf{x}), \tag{17.11}$$

where c_{ij} represents certain coefficients, and I describes the index set of the active subset. The authors in [292] also suggest a sparse greedy approach by proposing a criterion to optimize, in order to find the best parameters and induced points:

$$\begin{aligned} E(\mathbf{C}) &= \sum_{i=1}^{n} \|k(\mathbf{x}_i, \mathbf{x}) - \hat{k}(\mathbf{x}_i, \mathbf{x})\|^2_{\mathcal{H}} \\ &= \text{tr}\mathbf{K} - 2\text{tr}(\mathbf{C}\mathbf{K_{UX}}) + \text{tr}(\mathbf{C}\mathbf{K_{UU}}\mathbf{C}^T). \end{aligned} \tag{17.12}$$

Here, the coefficients are organized in a matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$. Minimizing $E(\mathbf{C})$ yields $\mathbf{C}_{\text{opt}} = \mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}$. Consequently, we achieve the approximation:

$$\mathbf{K_{XX}} \approx \mathbf{C}_{\text{opt}}\mathbf{K_{UX}} = \mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\mathbf{K_{UX}}, \tag{17.13}$$

where the induced points $\mathbf{U}$ are selected as a subset of the training data $\mathbf{X}$.

Unifying view

The authors in [268] provide a unifying perspective on aGPs. They interpret the algorithms as *exact inference with an approximate prior*. This approach offers an advantage in that it enables to compare different methods through the lens of various assumptions on the prior. The set of M latent variables, defined as $\mathbf{u} = [u_1, ..., u_M]^T \in \mathbb{R}^M$, is termed *inducing variables*, while the corresponding set of input locations $\mathbf{U} \in \mathbb{R}^{M \times d_{\text{det}}}$ is referred to as inducing inputs or induced points, as earlier mentioned. We are now interested in finding ways to modify the joint prior $p(\mathbf{y}, \mathbf{f}^*)$ in a manner that reduces computational cost.

For the induced points, we denote $p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|0, \mathbf{K_{UU}})$ as prior. We can recover the joint of a GP as given in Eq. 7.5, by marginalizing $\mathbf{u}$ from the joint GP prior

$$p(\mathbf{y}, \mathbf{f}^*) = \int p(\mathbf{y}, \mathbf{f}^*, \mathbf{u}) d\mathbf{u} = \int p(\mathbf{y}, \mathbf{f}^*) p(\mathbf{u}) d\mathbf{u}. \tag{17.14}$$

The authors introduce the major assumption of conditional independence between $\mathbf{y}$ and $\mathbf{f}^*$ given $\mathbf{u}$, which results in

$$p(\mathbf{y}, \mathbf{f}^*) \approx q(\mathbf{y}, \mathbf{f}^*) = \int q(\mathbf{y}|\mathbf{u}) q(\mathbf{f}^*|\mathbf{u}) p(\mathbf{u}) d\mathbf{u}. \tag{17.15}$$

Applying the properties for Gaussian distributions as given in Appendix, Eq. (B.17), on the joint distribution of $\mathbf{y}, \mathbf{f}^*$ and $\mathbf{u}$

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}(\mathbf{u}, \mathbf{y}, \mathbf{f}^* | 0, \begin{bmatrix} \mathbf{K_{UU}} & \mathbf{K_{UX}} & \mathbf{K_{UX^*}} \\ \mathbf{K_{XU}} & \mathbf{K_{XX}} + \sigma_\mathbf{n}{}^2\mathbf{I} & \mathbf{K_{XX^*}} \\ \mathbf{K_{X^*X}} & \mathbf{K_{UX^*}} & \mathbf{K_{X^*X^*}} \end{bmatrix}), \tag{17.16}$$

we derive the exact expressions for the two conditionals of Eq. (17.15):

- training:

$$p(\mathbf{y}|\mathbf{u}) = \mathcal{N}(\mathbf{y}|\mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\mathbf{u}, (\mathbf{K_{XX}} + \sigma_\mathbf{n}{}^2\mathbf{I}) - \mathbf{Q_{XX}}). \tag{17.17}$$

- test:

$$p(\mathbf{f}^*|\mathbf{u}) = \mathcal{N}(\mathbf{f}^*|\mathbf{K_{X^*U}}\mathbf{K_{UU}^{-1}}\mathbf{u}, \mathbf{K_{X^*X^*}} - \mathbf{Q_{X^*X^*}}). \tag{17.18}$$

It is noteworthy that both equations Eq. (17.17) and Eq. (17.18) include the term $\mathbf{K} - \mathbf{Q}$ in their covariance function. This can be understood as indicating the amount of information that $\mathbf{Q}$—and hence, the induced variables $\mathbf{u}$—contributes to $\mathbf{y}$ and $\mathbf{f}^*$, respectively. In terms of computational cost, the most complex operation now becomes the matrix-matrix product $\mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}$, with a complexity of $\mathcal{O}(NM^2)$, as the inversion over $\mathbf{K_{UU}}$ scales with the number of induced points M at a rate of $\mathcal{O}(M^3)$. Since we assume that $M << N$, this offers a great lever for computational cost reduction. The study by [268] provided a unifying interpretation for various aGP methods by demonstrating that most methods differ only in how they approximate these test and training conditionals.

As stated above, the topic of aGPs has been developed in the last decade and a variety of methods and extensions has been developed since: the authors of [193] use the spectral representation of the GP for sparsification. Vanhatalo and Vehtari [311] employ compact support kernels, which quickly decrease to zero for a sparse representation of the GP. The authors of [126, 156] discuss variational inference approaches. Wilson et al. [325] presents massively scalable GPs, leveraging grid structure of the training points and the resulting structure of the kernel matrix for better performance. The authors of [41, 42] show ways of updating aGPs when new training data arrives during training. Cao et al. [59] discuss different ways of optimizing the hyperparameters, including the induced variables. The authors of [63] present ways of comparing aGP methods. A review about recent developments is given in [208].

Implementation in SNOWPAC

We implemented three different aGPs in SNOWPAC that we are going to summarize in this section. The first was presented by [255, 313] as the *Subset of Regressor* (SOR) method. It uses the following priors for the training and test data:

- training:

$$q_{\mathrm{SOR}}(\mathbf{y}|\mathbf{u}) = \mathcal{N}(\mathbf{y}|\mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\mathbf{u}, \sigma_\mathbf{n}{}^2\mathbf{I}). \tag{17.19}$$

- test:

$$q_{\mathrm{SOR}}(\mathbf{f}^*|\mathbf{u}) = \mathcal{N}(\mathbf{f}^*|\mathbf{K}_{\mathbf{X}^*\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{u}, \mathbf{0}). \tag{17.20}$$

It has the predictive distribution

$$\begin{aligned} q_{\mathrm{SOR}}(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}^*|&\mathbf{Q}_{\mathbf{X}^*\mathbf{X}}(\mathbf{Q}_{\mathbf{XX}} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{y}, \\ &\mathbf{Q}_{\mathbf{X}^*\mathbf{X}^*} - \mathbf{Q}_{\mathbf{X}^*\mathbf{X}}(\mathbf{Q}_{\mathbf{XX}} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{Q}_{\mathbf{XX}^*}). \end{aligned} \tag{17.21}$$

The issue with SOR is that the predictive variance will tend to zero for points far away from the inducing points (assuming the used kernel tends to zero for points far apart).

Following this line of thought, the *Deterministic Training Conditional* (DTC) method was proposed by [284], building on the initial work by [199]. The primary concept is to utilize the exact prior for the test conditional, instead of employing approximated conditional priors for both the training and test conditionals. Below, we outline the training and test priors:

- training:

$$q_{\mathrm{DTC}}(\mathbf{y}|\mathbf{u}) = \mathcal{N}(\mathbf{f}|\mathbf{K}_{\mathbf{XU}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{u}, \sigma_{\mathbf{n}}{}^2\mathbf{I}). \tag{17.22}$$

- test:

$$q_{\mathrm{DTC}}(\mathbf{f}^*|\mathbf{u}) = p(\mathbf{f}^*|\mathbf{u}) = \mathcal{N}(\mathbf{f}^*|\mathbf{K}_{\mathbf{X}^*\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{XX}} - \mathbf{Q}_{\mathbf{XX}}). \tag{17.23}$$

This results in the following predictive distribution:

$$\begin{aligned} q_{\mathrm{DTC}}(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}^*|&\mathbf{Q}_{\mathbf{X}^*\mathbf{X}}(\mathbf{Q}_{\mathbf{XX}} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{y}, \\ &\mathbf{K}_{\mathbf{X}^*\mathbf{X}^*} - \mathbf{Q}_{\mathbf{X}^*\mathbf{X}}(\mathbf{Q}_{\mathbf{XX}} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{Q}_{\mathbf{XX}^*}). \end{aligned} \tag{17.24}$$

By employing the exact covariance matrix for test prediction, the predictive variance does not decrease to zero for points distant from the induced points. Instead, it converges to the prior variance $\mathbf{K}_{\mathbf{X}^*\mathbf{X}^*}$. Here, $\mathbf{K}_{\mathbf{X}^*\mathbf{X}^*}$ supplants the term $\mathbf{Q}_{\mathbf{X}^*\mathbf{X}^*}$ from SOR. Since $\mathbf{K}_{\mathbf{X}^*\mathbf{X}^*} - \mathbf{Q}_{\mathbf{X}^*\mathbf{X}^*}$ is positive definite (see [269]), $\mathbf{K}_{\mathbf{X}^*\mathbf{X}^*}$ is larger than $\mathbf{Q}_{\mathbf{X}^*\mathbf{X}^*}$. The additional term is the covariance of the exact test conditional, which approaches the prior distribution if $\mathbf{x}^*$ is far from the induced points. Consequently, the predictive variance of DTC is never less than the variance provided by SOR.

In [293], the authors introduced a different approach named *Sparse Pseudo-input Gaussian process.* Unlike DTC, this method suggests a distinctive approximation of the training conditional, while retaining the exact test conditional. It was re-named *Fully Independent Training Conditional* (FITC), reflecting the assumption of complete conditional independence between the training function values. The training and test priors are presented below:

- training:

$$q_{\mathrm{FITC}}(\mathbf{y}|\mathbf{u}) = \mathcal{N}(\mathbf{f}|\mathbf{K}_{\mathbf{XU}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{u}, \mathrm{diag}[\mathbf{K}_{\mathbf{XX}} - \mathbf{Q}_{\mathbf{XX}}] + \sigma_{\mathbf{n}}{}^2\mathbf{I}). \tag{17.25}$$

- test:

$$q_{\text{FITC}}(\mathbf{f}^*|\mathbf{u}) = p(\mathbf{f}^*|\mathbf{u}) = \mathcal{N}(\mathbf{f}|\mathbf{K}_{\mathbf{X}^*\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{XX}} - \mathbf{Q}_{\mathbf{XX}}). \tag{17.26}$$

Therefore, the predictive distribution is quite similar to that of DTC, apart from the adjustment made by the diagonal matrix,

$$\begin{aligned} q_{\text{FITC}}(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}^*|&\mathbf{Q}_{\mathbf{X}^*\mathbf{X}}(\mathbf{Q}_{\mathbf{XX}} + \mathbf{\Lambda}_{\text{FITC}} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{y}, \\ &\mathbf{K}_{\mathbf{X}^*\mathbf{X}^*} - \mathbf{Q}_{\mathbf{X}^*\mathbf{X}}(\mathbf{Q}_{\mathbf{XX}} + \mathbf{\Lambda}_{\text{FITC}} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{Q}_{\mathbf{XX}^*}), \end{aligned} \tag{17.27}$$

where $\mathbf{\Lambda}_{\text{FITC}} = \text{diag}[\mathbf{K}_{\mathbf{XX}} - \mathbf{Q}_{\mathbf{XX}}]$.

Over the years, FITC has emerged as the preferred method, sparking a multitude of improvements and variations: the authors of [235] introduce an efficient implementation of the algorithm, particularly with respect to hyperparameter estimation and the calculation of the marginal likelihood. In another advancement, Walder et al. [314] propose adding more flexibility to FITC by suggesting the use of different length scales $l_i, i = 1, ..., d_{\text{det}}$, for each induced point in the computation. The most recent development in FITC is the online adaptation proposed by Biil et al. in [42]. In their approach, they reformulate the training and evaluation of the predictive distribution to facilitate the addition of training points in an online manner, eliminating the need to rebuild the full kernel matrices $\mathbf{K}_{\mathbf{XX}}$ and $\mathbf{Q}_{\mathbf{XX}}$ each time. Furthermore, in a subsequent publication [41], they demonstrated ways to add induced points online under the condition of noisy training points.

Induced Points

Significant research has been invested in identifying the optimal subset of induced points. Smola et al. [291] propose incrementally expanding the index set of induced points through a greedy approach that minimizes the negative approximate maximum a posteriori estimate. The authors of [284] suggest a different error criterion by approximating the marginal likelihood. They demonstrate that their algorithm attains the speed of a random subselection of the induced points while yielding superior estimation accuracy.

Snelson et al. [293] suggest that the induced pair $(\mathbf{U}, \mathbf{u})$ need not be a subset of the training points $\mathbf{X}$. To demonstrate this, they designate $\mathbf{U}$ and $\mathbf{u}$ as pseudo-inputs and pseudo-targets, respectively. As $\mathbf{u}$ does not represent a real observation, they are also presumed to be noise-free. The only assumption about the induced points is a Gaussian prior, as already employed in previous sections:

$$p(\mathbf{u}|\mathbf{U}) = \mathcal{N}(\mathbf{u}|0, \mathbf{K}_{\mathbf{UU}}). \tag{17.28}$$

In observing the posterior distribution, such as Eq. (17.24) and Eq. (17.27), we see that it solely depends on the induced points $\mathbf{U}$, with no reliance on the induced variables $\mathbf{u}$. Consequently, Snelson et al. [293] also suggest to allow the induced points to exist in a continuous domain, as opposed to the discrete space of the training points subset. By adopting this assumption, the induced points can serve as additional hyperparameters in an optimization problem over the marginal likelihood, which we will explore in the next section.

Hyperparameter estimation for approximate Gaussian processes

A significant question that emerges is how to determine the optimal location for the induced points? Although the authors of [284] and [291] propose a greedy approach and select the induced points from the training set, we now have the opportunity to optimize over a continuous domain. In the following section, we will present the process of discovering appropriate induced points through optimization of the hyperparameters.

Besides the design parameters from the full GP parameter optimization,

$$\boldsymbol{\psi} = [\sigma_f, l_1, ..., l_{d_{\text{det}}}] \in \mathbb{R}_+^{(d_{\text{det}}+1)}, \tag{17.29}$$

we also incorporate the M induced points from $\mathbf{U}$ as design parameters. Consequently, this yields the following expanded parameter vector

$$\boldsymbol{\psi}_{\text{aGP}} = [\sigma_f, l_1, ..., l_{d_{\text{det}}}, \mathbf{U}] \in \mathbb{R}\,+^{(d_{\text{det}}+1)+Md_{\text{det}}}\,. \tag{17.30}$$

The authors in [293] use a single constant length-scale vector $\mathbf{l} = [l_1, ..., l_{d_{\text{det}}}]^T$, whereas Walder et al. [314] propose to assign each induced point $\mathbf{u}_i$ its own length-scale vector $\mathbf{l}_i$ for calculating the kernel matrix $\mathbf{K_{UX}}$. We adhere to the conventional approach by [293] as it introduces less dimensions (and, thus, less complexity) to the optimization problem.

To find the optimal hyperparameters, we employ the same approach as for the full GP and the adaptive kernels: we solve for the negative marginal log-likelihood given as

$$\min_{\boldsymbol{\psi}_{\text{aGP}}} -\log p(\mathbf{y}|\mathbf{X}) = \min_{\boldsymbol{\psi}_{\text{aGP}}} \frac{1}{2}\log|\mathbf{Q_{XX}} + \sigma_{\mathbf{n}}{}^2\mathbf{I}| + \frac{1}{2}\mathbf{y}^T(\mathbf{Q_{XX}} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{y}. \tag{17.31}$$

Following the unifying view by [268], we can use $\mathbf{Q_{XX}}$ and adapt it to the given low rank approximation approach:

- Full GP: $\mathbf{Q_{XX}} = \mathbf{K_{XX}}$.
- SOR & DTC: $\mathbf{Q_{XX}} = \mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\mathbf{K_{UX}}$.
- FITC: $\mathbf{Q_{XX}} = \mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\mathbf{K_{UX}} + \Lambda_{\text{FITC}}$.

Substituting the low rank approximation into the optimization problem, Eq. (17.31), we can leverage its structure to simplify the problem. We present the derivation for

FITC, but it is similar for SOR and DTC:

$$\begin{aligned}\min_{\psi_{\mathrm{aGP}}} -\log p(\mathbf{y}|\mathbf{X}) &= \frac{1}{2}\log\left|\mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\mathbf{K_{UX}} + \mathbf{\Lambda}_{\mathrm{FITC}} + {\sigma_{\mathbf{n}}}^2\mathbf{I}\right| \\ &\quad + \frac{1}{2}\mathbf{y}^T\underbrace{\left(\mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\mathbf{K_{UX}} + \mathbf{\Lambda}_{\mathrm{FITC}} + {\sigma_{\mathbf{n}}}^2\mathbf{I}\right)}_{\mathbf{\Gamma}}{}^{-1}\mathbf{y} \\ &= \frac{1}{2}\log\Big(|\mathbf{\Lambda}_{\mathrm{FITC}} + {\sigma_{\mathbf{n}}}^2\mathbf{I}| \cdot |\mathbf{K_{UU}^{-1}}| \cdot |\mathbf{K_{UU}} \\ &\quad + \mathbf{K_{UX}}(\mathbf{\Lambda}_{\mathrm{FITC}} + {\sigma_{\mathbf{n}}}^2\mathbf{I})^{-1}\mathbf{K_{XU}}|\Big) + \frac{1}{2}\mathbf{y}^T\mathbf{\Gamma}^{-1}\mathbf{y} \\ &= \frac{1}{2}\log|\mathbf{\Lambda}_{\mathrm{FITC}} + {\sigma_{\mathbf{n}}}^2\mathbf{I}| - \frac{1}{2}\log|\mathbf{K_{UU}}| \\ &\quad + \frac{1}{2}\log|\mathbf{K_{UU}} + \mathbf{K_{UX}}(\mathbf{\Lambda}_{\mathrm{FITC}} + {\sigma_{\mathbf{n}}}^2\mathbf{I})^{-1}\mathbf{K_{XU}}| + \frac{1}{2}\mathbf{y}^T\mathbf{\Gamma}^{-1}\mathbf{y}.\end{aligned} \tag{17.32}$$

Employing the Woodbury-Morris formula Eq. (B.1.1) for inversion,

$$\begin{aligned}\mathbf{\Gamma}^{-1} &= (\mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\mathbf{K_{UX}} + \mathbf{\Lambda}_{\mathrm{FITC}} + {\sigma_{\mathbf{n}}}^2\mathbf{I})^{-1} \\ &= (\mathbf{\Lambda}_{\mathrm{FITC}} + {\sigma_{\mathbf{n}}}^2\mathbf{I})^{-1} - (\mathbf{\Lambda}_{\mathrm{FITC}} + {\sigma_{\mathbf{n}}}^2\mathbf{I})^{-1} \\ &\quad \cdot \mathbf{K_{XU}}(\mathbf{K_{UU}^{-1}} + \mathbf{K_{UX}}(\mathbf{\Lambda}_{\mathrm{FITC}} + {\sigma_{\mathbf{n}}}^2\mathbf{I})^{-1}\mathbf{K_{XU}})^{-1}\mathbf{K_{UX}} \\ &\quad \cdot (\mathbf{\Lambda}_{\mathrm{FITC}} + {\sigma_{\mathbf{n}}}^2\mathbf{I})^{-1},\end{aligned} \tag{17.33}$$

this again results in an inversion of a M by M matrix instead of a N by N matrix.

We can also compute the gradients of the objective function with respect to each design parameter analytically. This provides valuable information for the optimization process, especially considering the high dimensionality of the problem. Specifically for aGPS, this now yields an even higher dimensional optimization problem due to the induced points. Though the individual operations to compute the gradients are straight-forward, they are still quite involved and technical. Thus, we present the different formulations in App. B.5.1 and B.5.2 for completeness.

Hock-Schittkowski Benchmark

In this section, we are going to employ SNOWPAC as a solver for the same benchmarks as before from Chap. 16—specifically, TP $\in \mathcal{TP} = \{29, 43, 100, 113, 227, 228, 268\}$—now contrasting the previously outlined GP methods. We are focusing on the problem formulation which seeks to minimize the expectation of the objective function while maintaining feasibility in expectation (see Eq. (16.2)), and the setup remains the same as before. For the sample estimators, we adopt varying sample sizes from the set $N \in \{200, 1000, 2000\}$. Additionally, for each GP method $S \in \{\mathrm{fullGP}, \mathrm{SOR}, \mathrm{DTC}, \mathrm{FITC}\}$, we carry out 100 optimization runs. We compare the relative error in the objective values, the constraint violations, and the discrepancies in the approximated optimal design relative to the deterministic solution. The sole exception is test problem TP = 268, for which we exhibit the absolute error for the objective function, since the optimum is 0.

The resulting plots are given in Fig. 17.5 and Fig. 17.6, with one row corresponding to one test problem. The test problem number is given in the title. They display the error in objective values, the approximated optimal design, and constraint violations from left to right. Furthermore, each set of three boxplots for each method is organized in ascending order of sample size N. Once again, these plots correspond to the first problem formulation. The boxplots for the second OUU formulation, Eq. (16.3), yielded similar outcomes and can be accessed in App. D.5.

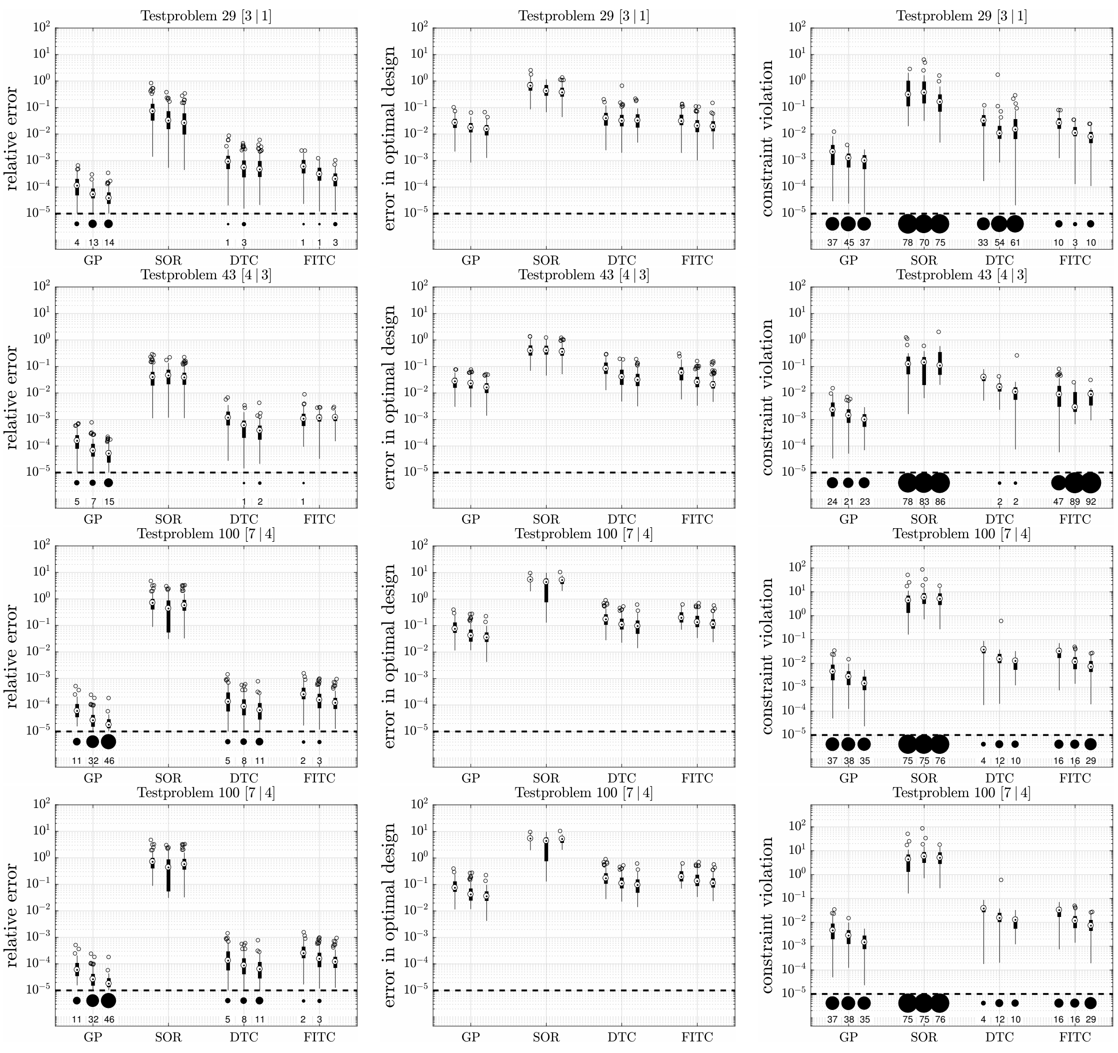


Figure 17.5: SNOWPAC Schittkowski benchmark for test problem TP = 29, TP = 43, TP = 100 and TP = 113 for relative error of the objective function, absolute error in optimal design and the constraint violations from left to right.

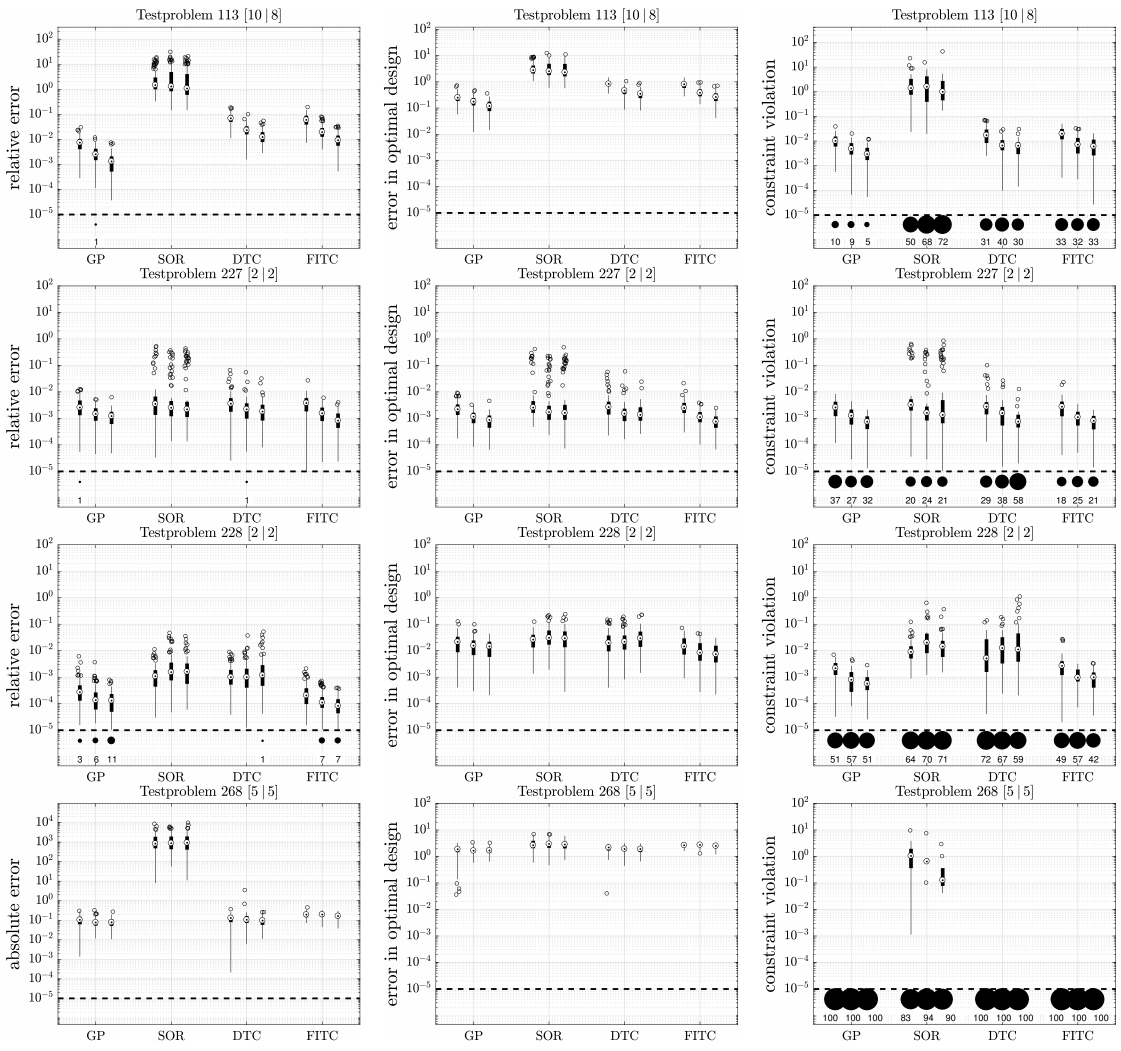


Figure 17.6: SNOWPAC Schittkowski benchmark for test problem TP = 227, TP = 228 and TP = 268 for relative error of the objective function, absolute error in optimal design and the constraint violations from left to right.

These findings align with our earlier observations. As expected, the full GP naturally delivers the best outcomes. The Subset of Regressors, on the one hand, appears to struggle the most in finding a solution within the given number of steps due to its inherent limitation. Nonetheless, it is most reliable in consistently staying within the feasible domain. On the other hand, the Fully Independent Training Conditional, being the most refined approximation method showcased here, displays the top results among the aGP techniques. The Deterministic Training Conditional falls between these two, a conclusion that aligns with its approximation technique. Still, it is noteworthy that all methods perform adequately even in higher dimensional scenarios. In this context, the weighting between the noisy sampling evaluations and the GP estimate plays a crucial role in maintaining a balance between the surrogates.

We again use the data profile presented by the authors in [232] to compare the different approaches over the number of iterations. We pick the tolerances as $\epsilon_f \in \{1\text{e-}2, 1\text{e-}3\}$ and $\epsilon_c \in \{1\text{e-}2, 1\text{e-}3\}$. We compare the exact GP with the SOR, DTC and FITC.

The data profiles depicted in Figs. 17.7 and 17.8 for both optimization problems Eq. (16.2) and Eq. (16.3) provide further support for the arguments mentioned earlier. Once again, the GP method demonstrates superior results by quickly finding an optimal solution. The second-fastest method is FITC, followed by DTC. SOR, on the other hand, exhibits the poorest performance, with a convergence rate of only about 48% for a tolerance of $\epsilon_f = \epsilon_c = 1\text{e-}2$. When employing a stricter tolerance of $\epsilon_f = \epsilon_c = 1\text{e-}3$, there is a general decrease in the number of runs that reach the optimum. It is worth noting that for small values of α, we observe a rapid initial descent in the GP, DTC, and FITC methods. This quick initial improvement is desirable for any optimization method, particularly when the black-box evaluations are costly.

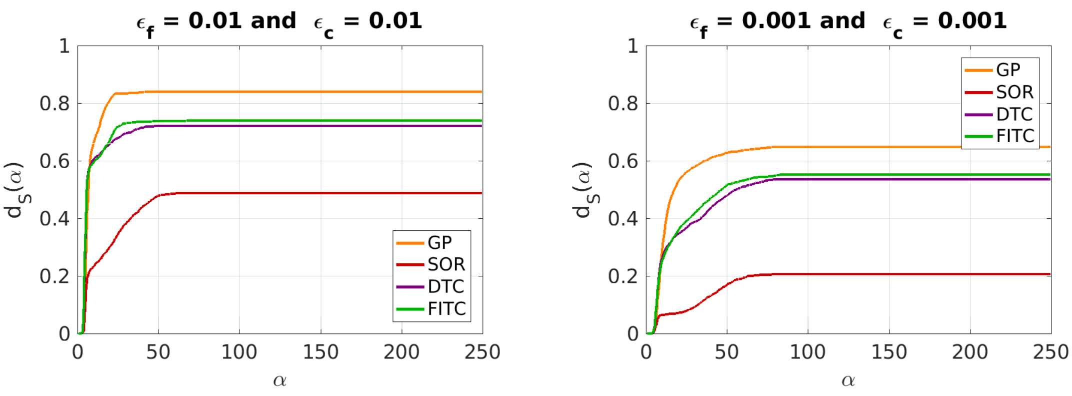


Figure 17.7: Data profile for the different methods and different error tolerances $\epsilon_f \in \{1\text{e-}2, 1\text{e-}3\}$ and $\epsilon_c \in \{1\text{e-}2, 1\text{e-}3\}$ for optimization problem Eq. (16.2).

We summarize the results, which we observe over all plots: across the array of problems, the exact GP consistently provides the best results, which we expected since it is the most accurate representation. The Subset of Regressors method tends to perform the worst, often by a factor of ten or even a hundred. The Deterministic Training Conditional method delivers better results than the Subset of Regressors approach utilizing

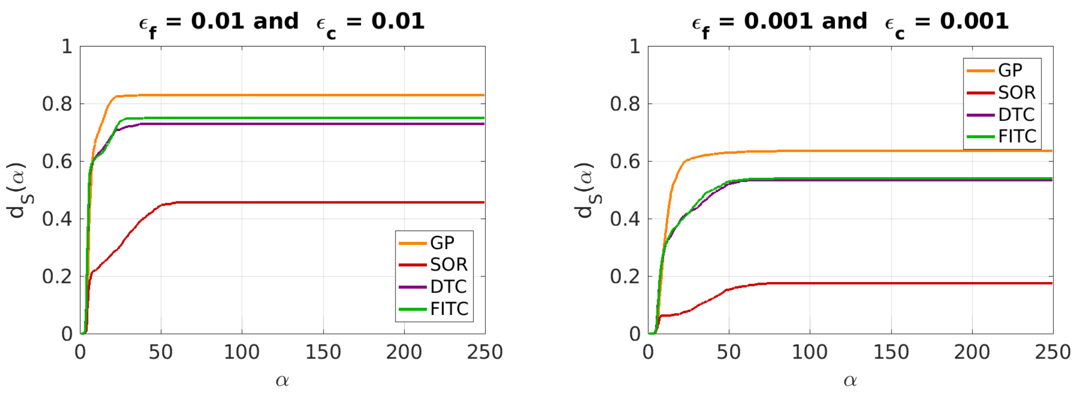


Figure 17.8: Data profile for the different methods and different error tolerances $\epsilon_f \in \{1\text{e-}2, 1\text{e-}3\}$ and $\epsilon_c \in \{1\text{e-}2, 1\text{e-}3\}$ for optimization problem Eq. 16.3.

the exact prior for test conditionals. Among the aGPs, the Fully Independent Training Conditional shows the best result with the assumption of independence between training values. Interestingly, in terms of constraint violations, the Subset of Regressors method provides the least infeasible solution. Finally, even though the error tends to grow with higher dimensions, all methods remain within acceptable error ranges and show a reasonable approximation quality.

Hence, the aGP methods implemented in SNOWPAC offer an alternative if computational expense of the GP evaluation becomes a bottleneck. This is possible, if we look at slowly converging problems where we have a high number of optimization iterations combined with cost efficient evaluation of the black-box problem itself. We still suggest to use the full GP as the standard surrogate for low numbers of evaluations or computationally expensive black-box applications since the inversion of the GP kernel matrix will not be the bottleneck in those cases. The presented methods can be activated in SNOWPAC by setting the option `GP_type` to `SOR`, `DTC` or `FITC`. The default value `GP` uses the full GP surrogate.

17.3 Mixed-integer optimization for SNOWPAC

Another use-case we envision for SNOWPAC is its application for optimization problems, where the optimization space not only encompasses continuous, but also discrete variables. This is the case for mixed-integer problems which can be formulated as:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathcal{R}^f(\mathbf{x}, \boldsymbol{\theta}), \\ \text{s.t.} \quad & \mathcal{R}^{c_i}(\mathbf{x}, \boldsymbol{\theta}) \leq 0, i = 1, ..., r. \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\ & \mathbf{x} = [\mathbf{x}_c^T, \mathbf{x}_d^T]^T \in \mathbb{R}^{d_{\det,c}} \times \mathbb{Z}^{d_{\det,d}}, \end{aligned} \tag{17.34}$$

where $\mathbf{x}_c^T$ and $\mathbf{x}_d^T$ denote our design in continuous and discrete space, respectively, such that $d_{\text{det},c} + d_{\text{det},d} = d_{\text{det}}$. The vectors $\mathbf{l} \in \mathbb{R}^{d_{\text{det}}}$ and $\mathbf{u} \in \mathbb{R}^{d_{\text{det}}}$ denote lower and upper box constraints.

We explore neural network architecture optimization as a useful field of applications: in its most basic form, a neural network consists of layers of different types, where each layer consists of neurons. The neurons between layers are connected with weights associated to each connection. In each neuron, a simple mathematical operation is performed and the weights weigh the importance of the neurons and the different layers. Through the connection of all those simple operations (and the ability to compute the analytic derivative of the operation), we aim to learn a target function. The learning takes place using available data, i.e., we try to learn a function $f(\mathbf{x})$, where we know $f(\mathbf{x}_i) = y_i$ for a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, and find the optimal weights for the connections through numerical optimization. We do not dive into details of neural networks here but refer to [46, 47, 196].

While the optimization for the best possible weights is in itself an intriguing task, which has sparked one of the most cited papers in science[2], we are more interested in how to actually design the neural network itself. We would like to know how to pick parameters, e.g., the right amount of layers, the right amount of neurons in each layer or the learning rate.

In the work that we explored in form of a master thesis with the student Kislaya Ravi [182], we have been working on an extension for SNOWPAC to be able to deal with such mixed-integer problems. The new demands required changes to the developed algorithm, which we will discuss next. The changes encompass adapting the trust-region shape and the surrogate optimization, and implementing a branch-and-bound algorithm for the solution of the integer design space. Finally, we will close this section with benchmarks results for a mixed-integer benchmark set and neural network architecture design for MNIST and CIFAR.

Trust-region shape

In SNOWPAC, the trust region takes the form of a sphere denoted as $B(\mathbf{x}_k, \rho_k) = \mathbf{x} \in \mathbb{R}^{d_{\text{det}}} : \|\mathbf{x} - \mathbf{x}_k\| \leq \rho_k$, with $\mathbf{x}_k$ as the center and ρ_k as the radius of the trust region. For mixed-integer problems, two kinds of design parameters are considered: integer and continuous parameters. The minimum permissible change in integer parameters is one, and must remain an integer. However, if the radius of the trust region (ρ_k) drops below 1, it becomes impossible to further alter the value of the integer parameters. As a result, the optimizer halts its exploration of other integer values, causing it to become stagnant. This implies that traditional trust-region mechanisms in SNOWPAC need to be modified if we aim for convergence to a local minimum. However, keeping the trust-region radius above one can prevent the algorithm from reaching convergence within the continuous space.

[2]The paper by Kingma and Ba [181] has 150894 citations based on Google Scholar as of the day of submission of this thesis.

As a solution, we adapt the trust region from spherical to box shape. We define it as:

$$B(\mathbf{x}_k, \rho_k) = \mathbf{x} \in \mathbb{R}^{d_{\text{det}}} : |\mathbf{x} - \mathbf{x}_k| \leq \rho_k. \tag{17.35}$$

Here, instead of a scalar, $\rho_k \in \mathbb{R}^{d_{\text{det}}}$, is a d_{det}-dimensional vector, where each dimension corresponds to the size of the box in each parameter dimension. For the integer dimensions, we set the minimum of the trust-region size to one. The maximum is limited by the upper and lower bounds, $\mathbf{u}$ and $\mathbf{l}$, of the problem.

We, therefore, integrate all changes and define ρ_k as:

$$\begin{aligned} \rho_k &= [\rho^{\mathbb{Z}^T}, \rho^{\mathbb{R}^T}]^T \in \mathbb{R}^{d_{\text{det},d}} \times \mathbb{R}^{d_{\text{det},c}}, \\ \text{where } \rho_i^{\mathbb{Z}} &\in [1, u_i - l_i) \forall i = 1, ..., d_{\text{det},d}, \\ \text{and } \rho_j^{\mathbb{R}} &\in (0, u_j - l_j) \forall j = 1, ..., d_{\text{det},c}. \end{aligned} \tag{17.36}$$

Surrogate optimization

Due to the changes of the optimization formulation and the trust region, we also have to update the surrogate optimization. Since the surrogate is quadratic, the mixed-integer problem belongs to the family of *Mixed Integer Quadratic Constrained Problems* (MIQCP) respectively. Instead of a continuous quadratic problem, we now have to solve a mixed-integer problem:

$$\begin{aligned} & \bar{\mathbf{x_k}} = \underset{\mathbf{x} \in B(x_k, \rho_k)}{\arg\min}\ m_k^{R^f}(\mathbf{x}) \\ \text{s.t.}\quad & m_k^{R^{c_i}}(\mathbf{x}) \leq 0 \qquad i = 1, 2, ..., r, \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\ & \mathbf{x} = [\mathbf{x}_c^T, \mathbf{x}_d^T]^T \in \mathbb{R}^{d_{\text{det},c}} \times \mathbb{Z}^{d_{\text{det},d}}. \end{aligned} \tag{17.37}$$

Several strategies exist for addressing this type of problem, as explored in depth in [29, 39, 43, 44]. The chosen methodology is reliant on the Hessian of the surrogate model. If the Hessian is positive definite, we can employ the commercial software solver CPLEX, as referenced in [49]. In cases where the Hessian is indefinite, but the $d_{\text{det},c}$-th principle leading sub-matrix is positive semidefinite, we can utilize the convex reformulation scheme detailed in [44].

If neither of these conditions are met, we can resort to the general *Branch and Bound* algorithm for problem resolution, as suggested in [238]. We have implemented this algorithm for the surrogate optimization of SNOWPAC. We are going to describe the idea of the algorithm and our implementation next.

Branch and Bound for surrogate optimization

The *branch-and-bound* technique is a frequently employed method for addressing $\mathcal{NP}$-hard combinatorial issues. It can be applied to a variety of problem types, including the Traveling Salesman problem, the Graph Partitioning problem, and the Quadratic

Assignment problem, as indicated in [71]. Initially, this approach was utilized to tackle Mixed Integer Linear Programming problems, as outlined in [90]. This methodology was subsequently expanded to non-linear problems, as documented in [51, 150, 300].

The fundamental concept of branch-and-bound algorithms revolves around utilizing the continuous relaxation of the mixed-integer problem at hand. After relaxing it to continuous, we can use standard continuous solvers to solve the problem. If the result of the optimization is integer and feasible, we consider the problem solved. Otherwise, in the branch step, we divide the problem at the current optimal point into two subproblem. For each subproblem, we assign distinct upper and lower integer bounds—the bound step. Next, we solve each subproblem individually and recursively repeat until we find a minimal integer solution. In the end, we receive an optimal design as for the continuous problem.

We, of course, have to do this for each integer dimension of the problem individually, which can result in a computationally expensive problem. However, we can also prune branches of the tree. This can happen, if, e.g. the lower bound of a sub-problem is larger than the upper bound of the current best solution. We summarize the algorithm in pseudo-code in Alg. 8.

Algorithm 8 Branch and Bound for mixed-integer problems

```
1: Input: $\mathbf{ub}$, $\mathbf{lb}$, $\mathbf{x}^*$, $f^*$
2: Bound: Create a subproblem as Eq. (17.37) with bounds $\mathbf{ub}$ and $\mathbf{lb}$
3: Relax integer criterion
4: Solve $\bar{\mathbf{x}} = \arg\min_{m_k^{c_i} \leq 0} m_k^f(\mathbf{x})$
5: Set local minimum of the branch($\bar{f}$) as $\bar{f} = m_k^f(\bar{\mathbf{x}})$
6: if $\bar{f} > f^*$ then
7:     Prune: Prune branch
8:     return $(\mathbf{x}^*, f^*)$
9: end if
10: if $\bar{\mathbf{x}}_j \in \mathbb{Z} \quad \forall \quad j = 1, ..., d_{\text{det},d}$ then
11:     Set $f^* = \bar{f}$
12:     Set $\mathbf{x}^* = \bar{\mathbf{x}}$
13:     return $(\mathbf{x}^*, f^*)$
14: end if
15: Select $l = \arg\max_{0 \leq j \leq d_{\text{det},d}} \ (|\bar{\mathbf{x}}_j| - floor(|\bar{\mathbf{x}}_j|))$
16: Set $\tilde{\mathbf{lb}} = \mathbf{lb}$ and $\tilde{\mathbf{ub}} = \mathbf{ub}$
17: Set $\tilde{\mathbf{lb}}_l = ceil(\bar{\mathbf{x}}_l)$
18: Branch: Recursively call Alg. 8 with $(\tilde{\mathbf{lb}}_l, \tilde{\mathbf{ub}}, \bar{\mathbf{x}}, \bar{f})$ for $(\mathbf{x}_r^*, f_r^*)$
19: Set $\tilde{\mathbf{ub}}_l = floor(\bar{\mathbf{x}}_l)$
20: Branch: Recursively call Alg. 8 with $(\tilde{\mathbf{lb}}, \tilde{\mathbf{ub}}_l, \bar{\mathbf{x}}, \bar{f})$ for $(\mathbf{x}_l^*, f_l^*)$
21: Compare: $b = \arg\min_{b \in \{l,r\}}(f_b^*)$
22: return $(\mathbf{x}_b^*, f_b^*)$
```

Compared to the original algorithm of SNOWPAC (see Alg. 7), our changes are twofold: first, we substitute the continuous surrogate optimization by the mixed-integer optimization, employing the branch-and-bound algorithm, Alg. 8. Second, the trust region itself is now box-shaped, not allowing ranges below one for the integer dimensions. The GP still operates as before, smoothing the function values, and the fully-linear surrogate model still present a continuous function.

Benchmark results

We assess and contrast the performance of our algorithm against other optimization methods using a collection of benchmark problems. Apart from SNOWPAC, the compared solvers include pySOT [112], TPE (Tree-Structured Parzen's Estimator) [34, 35], SMAC (Sequential Model-based Algorithm) [164, 165], and NOMAD [89, 195].

The benchmark problems used for comparison are detailed in Table 17.1, their complete problem statements can be found in App. D.6.1. We categorize these benchmarks into two types: the first type comprises *Sphere* problems, which are convex functions with a single unique minimum. This set includes five problems, each with a progressively increasing number of design parameters. This category is used to analyze, how varying the dimensionality impacts the behavior of the different optimization tools. The second category contains more challenging optimization problems. Some problems in this group are multi-modal, meaning they have multiple local minima, while others feature elongated axes for certain design parameters. By using this category of optimization problems, we are able to assess the ability of various tools to handle specific complex scenarios.

Sr.No.	**Category**	**Problem Name**	**n**	$\mathbf{n_d}$	$\mathbf{n_c}$
1	Category 1	Sphere Problem [333]	2	1	1
2			4	2	2
3			6	3	3
4			8	4	4
5			10	5	5
6	Category 2	Ackley's Function [10, 298, 333]	8	3	5
7		De Jong's Problem [333]	5	3	2
8		Bohachevsky Function 1 [10, 48]	2	1	1
9		Bohachevsky Function 2 [10, 48]	2	1	1
10		Griewank Function [333]	10	5	5

Table 17.1: List of benchmark problems used for comparison of optimization tools.

Following the pattern of previous benchmark evaluations, we employ data profile plots to aggregate the results. Each benchmark problem is solved 10 times, and the profile graphs are plotted considering two accuracy thresholds, namely ϵ_f = 1e-1 and ϵ_f = 1e-2. In each experiment series, we select a random integer starting point within the trust-region. For SNOWPAC, an initial box-dimension is also necessary, which we specify. We start our results with the deterministic case before we move to a stochastic benchmark test. Additionally, we resort to unconstrained cases.

Fig. 17.9 presents the performance profile for the first category of benchmark problems. Among all optimizers, SNOWPAC (depicted by the blue line) delivers the best performance. This is closely followed by pySOT (red line), NOMAD (purple line), TPE (orange line), and SMAC (green line). The performance of SNOWPAC and pySOT is notably superior to the other tools, a fact attributable to their trust-region derivative-free optimization method foundation. These tools rapidly converge towards a local minimum.

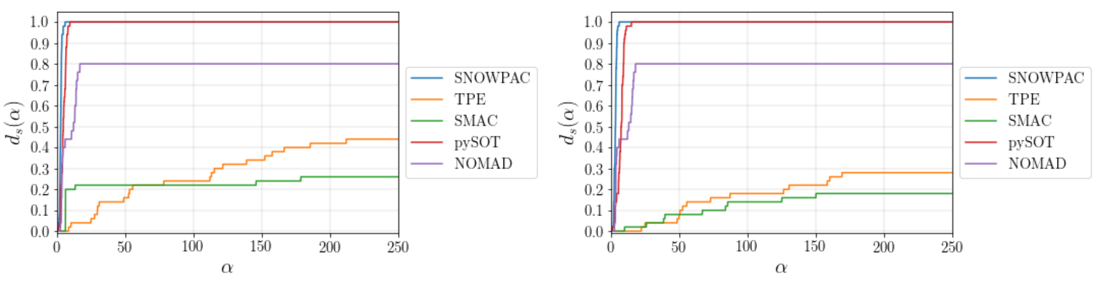


Figure 17.9: Data profile for the first category of Table 17.1 for ϵ_f = 1e-1 (left) and ϵ_f = 1e-2 (right).

SNOWPAC outperforms pySOT for smaller values of α, as it begins optimization with a fully linear model, which necessitates at least $n+1$ evaluation points, with n being the number of design parameters. Conversely, pySOT requires $2n+1$ points. This means SNOWPAC starts searching for trial points ahead of pySOT, explaining its improved performance for low α.

NOMAD is a directional search method and its performance is less impressive compared to trust-region methods, as reflected in [239]. SMAC and TPE are methods based on Bayesian optimization, with a tree structure added to expedite the calculation of trial nodes [34, 165]. Bayesian Optimization underperforms compared to the trust-region methods for continuous constrained problems, which is also observable in Fig. 17.9.

Fig. 17.10 presents the performance profile of the benchmarked optimization tools across the second category of benchmark problems. As these optimization problems are more challenging to solve than those in the first category, a decline in performance is observed. Compared to the others, SNOWPAC ranks the highest. Meanwhile, the performance profiles of TPE and SMAC are observed to be the lowest, primarily because they are both grounded in Bayesian optimization.

Bayesian optimization typically underperforms in optimizing multi-modal functions because it generates surrogates for the entire design space. Given the highly oscillatory nature of multi-modal functions, many points are needed to generate a global surrogate. Due to the inaccuracy of the surrogate, a substandard performance is observed from TPE and SMAC.

Unlike the Bayesian optimization method, trust-region-based methods construct only a local surrogate. An accurate local surrogate model can be assembled with just a few

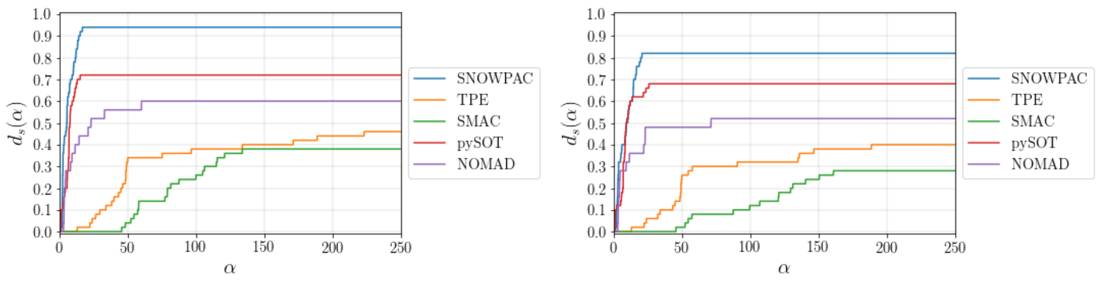


Figure 17.10: Data profile for the second category of Table 17.1 for ϵ_f = 1e-1 (left) and ϵ_f = 1e-2 (right).

points. As a result, trust-region-based optimization methods converge swiftly towards a local minimum. However, for multi-modal functions, multiple local minima exist, and it is not guaranteed to converge to the global minimum. If the global minimum is dominant, i.e., there is a significant difference between function values from local minima, the chances of finding the global minimum increase. Dewancker et al. provide a detailed comparison of various Bayesian optimization tools for multi-modal functions in [97].

To close this section on benchmark results, we delve into the performance comparison between the optimization tools when applied to stochastic functions. We only consider the first category of benchmark problems, specifically the Sphere Problem [333], to which we add a uniform noise. We take the expected value, $\mathcal{R}_0$ as the robustness measure that needs to be optimized. To estimate the measure, we employ a sample size of 100.

Fig. 17.11 presents the performance profile graph for stochastic sphere functions. The observed behavior parallels that of Fig. 17.9. The diminished performance of TPE and SMAC is due to the fact that Bayesian optimization tends to underperform with high-dimensional problems. Both SNOWPAC and pySOT deliver comparable performance, which can be attributed to their common basis in the trust-region method. Additionally, it is worth pointing out that although convergence appears somewhat slower (as evidenced by the rightward shift of the profile curve), both SNOWPAC and pySOT achieve similar accuracy to that seen in the deterministic case.

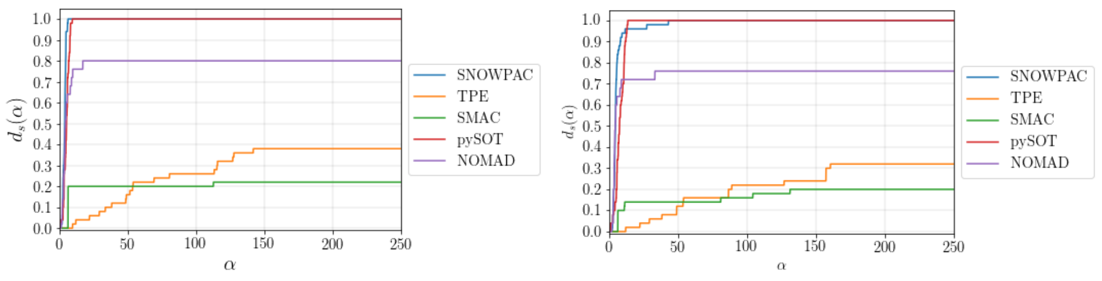


Figure 17.11: Data profile for stochastic sphere functions for ϵ_f = 1e-1 (left) and ϵ_f = 1e-2 (right).

Neural network optimization

At the start of this chapter, we pointed out that the optimization of a neural network architecture presents a significant challenge for mixed-integer optimization. As such, we next put SNOWPAC to the test in this context: we establish four distinct setups to compare various optimizers, ensuring the hyperparameters are scaled to be of similar magnitude. The specifics of the scaling can be found in Tables D.1-D.4. We will now delve into the fundamental structure of the four problems.

The problem design for this experiment draws inspiration from the work of Ilievski et. al. [166]. In the first problem set, we optimize six hyperparameters of a Multilayer Perceptron (MLP) network. Out of these six hyperparameters, two are integer parameters while the remaining four are continuous. A detailed list of the optimized hyperparameters can be found in Table D.1. This problem will be referred to as **6-MLP** in the subsequent discussions.

The MLP network comprises two hidden layers, with ReLU activation function in between and a SoftMax loss term. Stochastic Gradient Descent (SGD) is employed for training the weights of the neural network. The network's objective is to classify grayscale images of handwritten digits from the renowned benchmark dataset, *MNIST* [197].

MNIST is a database composed of handwritten digits represented as grayscale images. Each image is labeled with a number ranging from 0 to 9, based on the depicted digit. The dataset contains 60,000 training samples and 10,000 testing/validation samples. The handwritten digits are size-normalized and centered in a fixed-size image, with each image measuring 28 × 28 pixels. A few sample images from the MNIST dataset are displayed on the left side of Fig. 17.12.

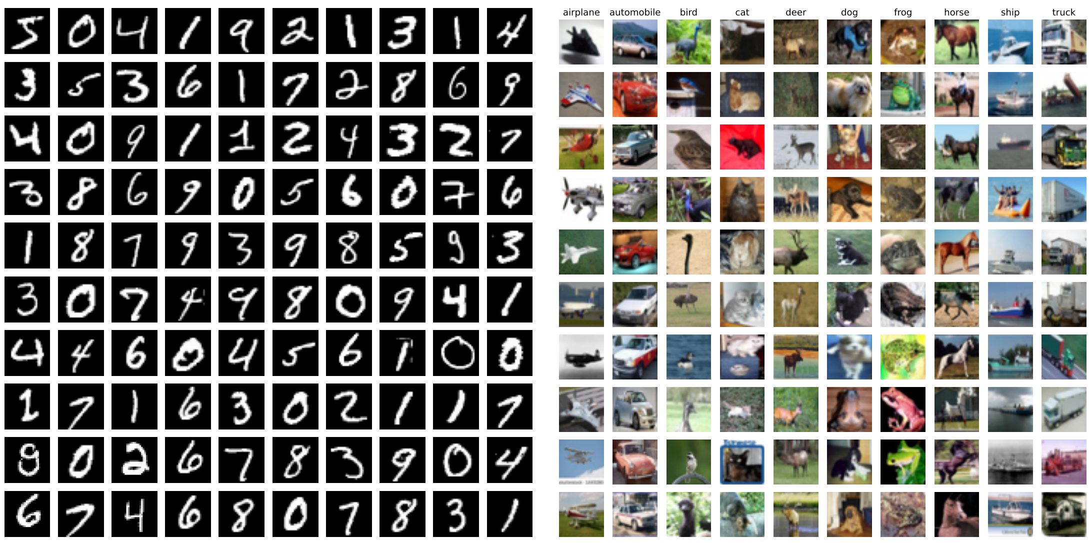


Figure 17.12: Sample images from the MNIST [197] (left) and CIFAR10 datasets [187] (right).

The second problem involves optimizing eight hyperparameters of a Convolutional Neural Network (CNN). This includes four integer hyperparameters and four continuous hyperparameters. The CNN architecture encompasses two convolutional blocks, each consisting of a convolutional layer with batch normalization, followed by ReLU activation and a 3×3 max-pooling layer. Following the convolutional blocks, we have two fully-connected layers with LeakyReLU activation, and a SoftMax loss term is used.

The purpose of this network is to classify images into digits, employing the MNIST dataset. We will refer to this problem as **8-CNN** in the following discussions. For a comprehensive breakdown of the optimized hyperparameters, please refer to Table D.2.

We follow the same format in the third problem. We just increase the number of target hyperparameters to be optimized to fifteen, composed of five integer parameters and ten continuous parameters. We will refer to this problem as **15-CNN** in subsequent discussions. For a detailed breakdown of the hyperparameters being optimized, please refer to Table D.3.

The fourth problem considers the *CIFAR-10* dataset. The network is trained to classify colored images. The CIFAR-10 dataset [187] contains 60000 color images, each measuring 32×32 pixels. It features 10 classes, with each class comprising 6000 images. The dataset includes 50000 training images and 10000 test images. Some sample images from each CIFAR-10 class are displayed on the right side of Fig. 17.12. It involves optimizing nineteen hyperparameters, which include five integer parameters and fourteen continuous parameters. We refer to this problem as **19-CNN** in further discussions. The architecture is identical to that of 8-CNN and 15-CNN, with the addition of dropout layers following the convolutional and fully-connected layers. For a detailed breakdown of the hyperparameters being optimized, we refer to Table D.4.

While training neural networks is well-suited for graphical processing units, it still demands significant computational resources. Consequently, it is advantageous to identify the optimal hyperparameter values within a restricted number of complete network trainings. Hence, we cap the number of optimization iterations at 200 black-box evaluations, where one black-box evaluation is equal to one full network training cycle. We conduct five runs of each setup, each time using a different random seed for neural network weight initialization. This random seed remains consistent across all methods.

We compare SNOWPAC against HORD [166], HORD-ISP [166] (both HORD and HORD-ISP uses pySOT [112]), Spearmint [87], SMAC [164, 165] and TPE [34, 35]. Certain optimizers require an initial starting point. We document these initial points in App. D.6.2. They represent a form of user-provided a priori information to the optimizer. The optimization tools that require these starting points include SNOWPAC, HORD-ISP, and SMAC. SNOWPAC also requires the specification of a starting trust-region dimension. We determine this value such that the entire hyperparameter space falls within the initial trust region. The starting trust-region box-dimension details are provided in Table D.5.

Table 17.2 shows the mean and the standard deviation of percentage validation error over five experiments. We observe that apart from MNIST 8-CNN, SNOWPAC reaches the smallest validation error for the different benchmarks, also for the high-dimensional problems.

	MNIST 6-MLP	MNIST 8-CNN	MNIST 15-CNN	CIFAR-10 19-CNN
SNOWPAC	**1.38 (0.03)**	0.87 (0.08)	**0.92 (0.13)**	**22.83 (1.27)**
HORD	1.45 (0.03)	0.89 (0.05)	1.1 (0.14)	23.41 (0.73)
HORD-ISP	1.45 (0.05)	1.01 (0.06)	1.35 (0.36)	24 (0.26)
Spearmint	1.44 (0.04)	**0.85 (0.07)**	1.03 (0.06)	23.85 (0.78)
TPE	1.58 (0.02)	1.11 (0.04)	1.58 (0.09)	26.4 (0.38)
SMAC	1.51 (0.05)	0.92 (0.02)	1.35 (0.12)	24.06 (0.92)

Table 17.2: Mean and standard deviation (inside parenthesis) of percentage validation error over 5 sample of experiments. We mark the best results for each case in bold.

We plot the number of function evaluations for the four distinct problems in relation to the relative validation error in Fig. 17.13. We note that SNOWPAC rapidly finds its optimal solution. This outcome is expected given that the loss-surface of a multilayer network is typically highly convex with numerous local minimums [69]; this leads us to anticipate a multi-modal objective for the neural network architecture as well. As stated before, trust-region based optimization methods are known to quickly converge to such local minima. SNOWPAC not only accomplishes this rapidly, but often finds the best solution with the lowest validation error.

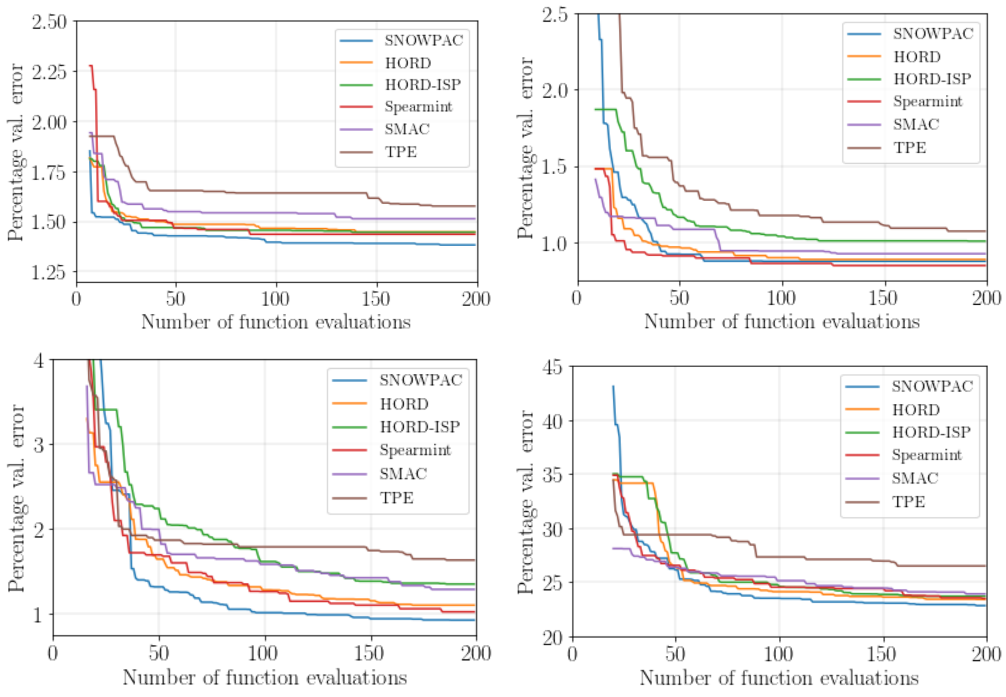


Figure 17.13: Average percentage validation error versus the number of function evaluations for all the setups in scaled hyperparameter space for 6-MLP (top left), 8-CNN (top right), 15-CNN (bottom left) and 19-CNN (bottom right).

We conclude that the mixed-integer extension of SNOWPAC holds up well against other methods in the field. This new extension paves the way for an intriguing area of new applications for the method. While we have been focused on unconstrained methods in these initial tests, we anticipate extending the method to tackle constrained problems, once again using the inner boundary path to sidestep infeasible solutions. Given its capability for OUU, we also see potential for its application in the pursuit of robust and reliable neural networks, as opposed to merely searching within a deterministic space.

Part V

Multilevel Monte Carlo for optimization under uncertainty

I pass with relief from the tossing sea of Cause and Theory to the firm ground of Result and Fact.

—*Winston Churchill [70]*

18 Connecting SNOWPAC with MLMC

As final question and third contribution in this work, we tackle the coupling of our first contribution from Part III with our second contribution of Part IV. We present how to use MLMC estimators in derivative-free OUU and answer the question if our newly developed estimators have an impact on the performance of the optimization. We already saw improvements regarding accuracy or computational cost for sampling alone in Part III and wonder if this directly translates to OUU.

We, thus, will present how we link the new MLMC estimators with SNOWPAC in this section. Afterwards, we will present results of this new approach in Chap. 19 on two benchmark problems and a realistic wake-steering scenario, where we designed multilevel cases. Both chapters of this part are based on work in [223] and [224].

We have discussed the algorithmic specifics of the MLMC resource allocation already in Section 12.6. Now, we want to use these new MLMC methods as estimators of the measures $\mathcal{R}_i, i = 1, 2$, in SNOWPAC and compare it to the standard MLMC estimator, $\mathcal{R}_0$, targeting the mean. For that, let us denote our newly presented MLMC estimators for higher/order moments in Part III following the notation of our measures in Section 9.1 for an objective or constraint function $b \in \{f, c_1, ..., c_r\}$. We juxtapose the notations (and the relevant equations) from the different chapters in Table 18.1.

Statistic	Measure (Eq.)	Estimator	MLMC estimator (Eq.)
$\mathbb{E}$	$\mathcal{R}_0^b$ (9.2)	R_0^b	$\widehat{\mu}_{1,\mathrm{ML}}^b$ (6.21)
$\mathbb{V}$	$(\mathcal{R}_1^b)^2$ (9.3)	$(R_1^b)^2$	$\widehat{\mu}_{2,\mathrm{ML}}^b$ (12.9)
σ	$\mathcal{R}_1^b$ (9.3)	R_1^b	$\widehat{\sigma}_{\mathrm{ML,biased}}^b$ (12.21)
$\mathbb{E} + \alpha\sigma$	$\mathcal{R}_2^b$ (9.4)	R_2^b	$\widehat{\zeta}_{\mathrm{ML,biased}}^b = \widehat{\mu}_{1,\mathrm{ML}}^b + \alpha\widehat{\sigma}_{\mathrm{ML,biased}}^b$ (12.25)

Table 18.1: Notations for the different statistics and estimators from OUU and MLMC.

When combining this new MLMC strategy for higher-order moments with SNOWPAC, we not only use the new estimator. It is also necessary to offer an estimation of the standard error for the generic objective R^f and constraint functions $\{R^{c_i}\}_{i=1}^r$. This estimate is used for the noise $\varepsilon^b, b \in \{f, c_1, ..., c_r\}$, where SNOWPAC employs $\varepsilon^b = t_\nu\sqrt{\mathbb{V}[R_i]}$ as noted in Section 14.1. As a reminder, we employ the noise as lower bound for the adjustment of the trust region, $\rho_k \geq \lambda_t\sqrt{\bar{\varepsilon}_{\max}^k}$, and when balancing of the evaluations with the GP: $\tilde{R}_k^b = \gamma_k\mathcal{G}_k^b[R_k^b] + (1-\gamma_k)R_k^b$. In our case, we have already calculated the variances of all new estimators because they are needed in their respective resource allocation problems (either as constraint or objective). Hence, in Table 18.2, we connect the variances of the MLMC estimators from Part III to the variance for the estimator of measures in SNOWPAC of Part IV.

Variance	MLMC estimator	Eq.
$\mathbb{V}[R_0^b]$	$\mathbb{V}[\widehat{\mu}_{1,\mathrm{ML}}^b]$	(6.24)
$\mathbb{V}[(R_1^b)^2]$	$\mathbb{V}[\widehat{\mu}_{2,\mathrm{ML}}^b]$	(12.11)
$\mathbb{V}[R_1^b]$	$\mathbb{V}[\widehat{\sigma}_{\mathrm{ML,biased}}^b]$	(12.23)
$\mathbb{V}[R_2^b]$	$\mathbb{V}[\widehat{\zeta}_{\mathrm{ML,biased}}^b]$	(12.27)

Table 18.2: Notations for the variance of the different estimators from OUU and MLMC.

We have two possibilities of using SNOWPAC with the new MLMC estimators coupled to the application code. The first possibility is the coupling as described in Section 15. To repeat, the black-box application is linked to SNOWPAC by implementing the interface of the `BlackBoxBaseClass` from SNOWPAC. Evaluating the objective and constraints is fully handled by the application. Hence, also the new MLMC estimators have to be implemented on the solver side and are not directly available in SNOWPAC. SNOWPAC only communicates the current design and expects a result plus a noise estimate for ε^b. We visualize this pipeline schematically in Fig. 18.1.

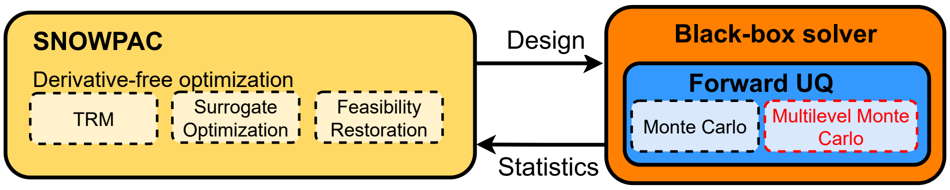


Figure 18.1: Coupling of general surrogate models for a black-box solver with SNOWPAC where the forward UQ evaluation is handled by the application.

The second possibility is using Dakota, where we implemented the new MLMC estimators and which can be linked to SNOWPAC. In this case, the outer loop of the optimization is orchestrated by Dakota. Dakota includes SNOWPAC as an externally-developed solver, whose settings can be chosen by the user through the input file. Within this setup, SNOWPAC deploys the previously mentioned trust-region management (TRM), surrogate optimization, and the feasibility restoration, as presented in Part IV. It communicates the subsequent design step to Dakota, which then organizes the forward UQ problem at the current design. Dakota takes charge of sampling, interacting with the black-box application, and gathering the results, utilizing the MLMC estimators as presented in Part III. Finally, the computed statistics for both the objective, constraints and noise estimates are transmitted to SNOWPAC for the ensuing optimization phase. A schematic depiction of how SNOWPAC, the forward UQ problem in Dakota, and the application of interest interact is shown in Fig. 18.2.

We can couple SNOWPAC with MLMC through the usage of `nested_models` in the Dakota input file. We choose `snowpac` as top-level method, where we can set the settings as described in Section 15. In the nested model, we denote the optimization problem by defining the accumulation of statistics through the `primary_`- and `secondary_response_mapping`. In the model, we also link to the submethod via

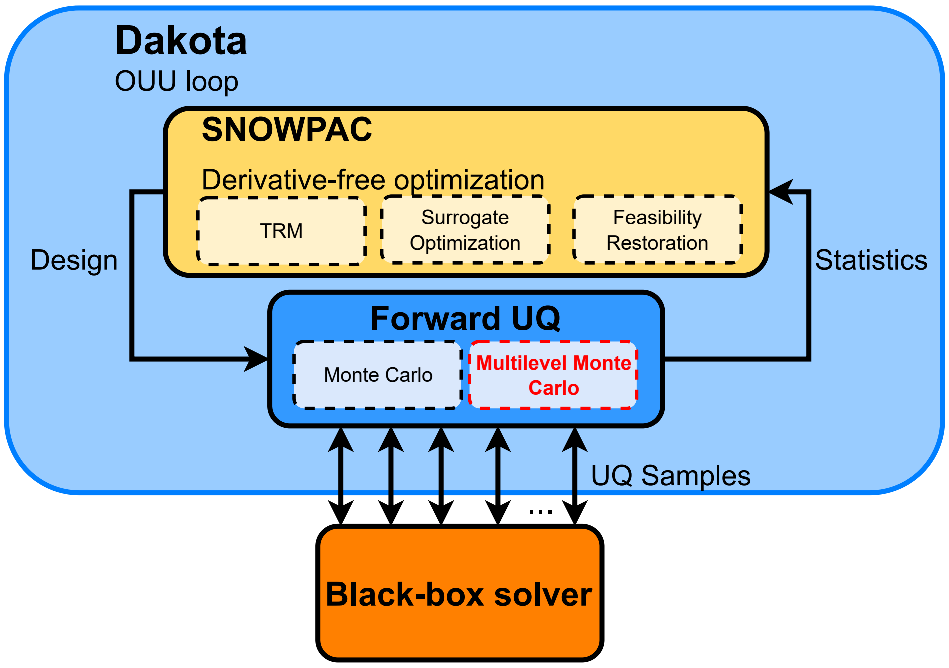


Figure 18.2: Coupling of general surrogate models for a black-box solver with SNOWPAC as outer-loop optimization method where the whole process is controlled by Dakota.

the `sub_method_pointer`. In the submethod block, we set `multilevel_sampling` as method, again with the settings as presented in Section 12.6. We give an example of the `nested_models` in the Dakota input file in Example 12.

Example 12 (Example for a nested model.)**.** *This nested model connects an optimization problem with one objective and two constraints with the underlying MLMC sample estimator. The parameters* `variables_-` *and* `response_pointer` *point to the input variables and output responses for the optimization problem. The* `sub_method_pointer` *points to the MLMC method. We define in the response mapping how to combine the returned responses. In this case, we have three response functions (one objective, two constraints), where we get a mean and a standard deviation statistic for each. In* `primary_response_mapping`*, which defines the objective, we, e.g., combine the mean of the first response with three times the standard deviation of the first response. Hence, this is a scalarization objective, $\mathcal{R}_2^f$, with $\alpha = 3$. For the* `secondary_response_mapping`*, we use the mean of the second response in the first mapping, $\mathcal{R}_0^{c1}$, and the mean of the third response in the second mapping, $\mathcal{R}_0^{c2}$. These correspond to our constraint functions.*

```
model,
      id_model = 'OPTIM_M'
      nested
         variables_pointer   = 'OPTIM_V'
         sub_method_pointer  = 'UQ'
         responses_pointer   = 'OPTIM_R'
```

```
        primary_response_mapping   = 1. 3. 0. 0. 0. 0.
        secondary_response_mapping = 0. 0. 1. 0. 0. 0.
                                     0. 0. 0. 0. 1. 0.
```

Listing 18.1: Extract of Dakota input file for SNOWPAC with MLMC.

All our upcoming results are computed using Dakota, since it offers a convenient way of coupling all our contributions. Dakota also provides parallelization, which is highly effective for MC and MLMC since the evaluations are embarrassingly parallel. We still remark that Dakota might be computationally disadvantageous compared to a direct link of SNOWPAC and the application, when the model is computationally cheap. Dakota communicates with the application through input and output files, where reading and writing the files can become the bottleneck. Nevertheless, this is not to be expected to be a common case apart from benchmark tests since these methods are specifically intended for computationally expensive problems.

19 Numerical results

Finally, we want to see how the new developments in MLMC sampling and derivative-free stochastic optimization coupled together perform in a full OUU pipeline. Whereas we have only regarded sampling in Part III, we now have to compute a sampling estimate in each OUU iteration of SNOWPAC. We focus on comparing the expected value with the scalarization as our new contribution.

In this chapter, we will present results for three different problems. First, we will revisit Problem 18 in Section 19.2 from Chap. 13. However, now we will focus on optimization instead of sampling only. Second, we will present results for the Rosenbrock function in Section 19.3, a popular optimization benchmark. Third, we will end this section with discussing a wake-steering scenario for wind power plants in Section 19.4. For all cases, we design new multilevel test cases and validate our results against standard MC sampling. Before we dive into the numerical results of both benchmarks, however, we will present the experimental setup and define distance metrics that we use to quantitatively assess our results in Section 19.1.

19.1 Experimental design and distance metrics

Prior to delving into the actual numerical results, we detail problem design and the error quantities utilized for a quantitative assessment of the OUU workflows. For the first two applications of Problem 18 in Section 19.2 and the Rosenbrock function in Section 19.3, we are able to (at least numerically) compute a reference solution by computing the target variance $\epsilon_{\mathbb{X}}^2$, where $\mathbb{X} \in \{\mathbb{E}, \mathbb{S}\}$. We pick as target the variance of a MC estimator using $N_L = 1000$ samples on the finest level L. Consequently, as in the sampling case, we anticipate the MLMC estimators to perform comparably to the MC estimator since it is aiming for the same accuracy in $\epsilon_{\mathbb{X}}^2$.

In order to get quantitative comparisons between the different approaches, we execute $M^{\mathbb{X}} = 25$ optimization runs for each method. A set of $M^{\mathrm{MC}} = 25$ independent MC optimization runs serve as our reference point. For each of these runs, we extract the final optimal design $\mathbf{x}_i^{\mathbb{X}} \in \mathbb{R}^{d_{\det}}$ and $\mathbf{x}_i^{\mathrm{MC}} \in \mathbb{R}^{d_{\det}}$ as identified after a fixed number of iterations of SNOWPAC. We assign $x_{i,j}^{\mathbb{X}}$ to denote the j-th element of $\mathbf{x}_i^{\mathbb{X}}$, and analogously for $x_{i,j}^{\mathrm{MC}}$. This yields sets of final designs for MLMC and MC, represented as $\mathbf{X}^{\mathbb{X}} = \{\mathbf{x}_i^{\mathbb{X}}\}_{i=1}^{M^{\mathbb{X}}}$ and $\mathbf{X}^{\mathrm{MC}} = \{\mathbf{x}_i^{\mathrm{MC}}\}_{i=1}^{M^{\mathrm{MC}}}$, respectively. We define the Euclidean distance between two vectors as $e(\mathbf{x}, \mathbf{y}) := \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$, the difference between two vectors as $d(\mathbf{x}, \mathbf{y}) := [|x_0 - y_0|, ..., |x_d - y_d|] \in \mathbb{R}^d$, the mean vector over a set as $\overline{\mathbf{X}} := \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i$ and the standard deviation value of a set as $\hat{\sigma}(\mathbf{X}) = \frac{1}{M-1}\sum_{i=1}^{M}(\mathbf{x}_i - \overline{\mathbf{X}})^2$.

Using these definitions, we formulate three metrics as follows:

- The first metric quantifies the distance between the centers of two sets of designs
$$\mathrm{Dist}_C^{\mathbb{X}} = e(\overline{\mathbf{X}}^{\mathbb{X}}, \overline{\mathbf{X}}^{\mathrm{MC}}).$$
- The second metric considers the distance in standard deviation values of the two sets
$$\mathrm{Dist}_\sigma^{\mathbb{X}} = d(\hat{\sigma}(\mathbf{X}^{\mathbb{X}}), \hat{\sigma}(\mathbf{X}^{\mathrm{MC}})).$$
We highlight here, that this is a d-dimensional metric, showcasing the variation in standard deviation across each dimension.
- Lastly, for our third metric, we utilize the root-mean-square deviation to the MC reference solution as follows:
$$\mathrm{Dist}_{\mathrm{RMSdev}}^{\mathbb{X}} = \sqrt{\frac{1}{M^{\mathbb{X}}} \sum_{i=1}^{M^{\mathbb{X}}} e(\mathbf{x}_i^{\mathbb{X}}, \overline{\mathbf{X}}^{\mathrm{MC}})^2}.$$

The last metric is a common measurement used in bioinformatics for estimating the average distance between atoms in proteins [86] or force differences in molecules [247], to cite a few examples. Generally, this distance is calculated in relation to a benchmark structure or an average atomic position. This metric is well-suited for our needs, since we are interested in the variance between cluster of points with respect to a reference solution. Additionally, it can be interpreted as a combination of the previous two metrics.

Using all three metrics, we are able to quantify the closeness of point clouds. In our case, we will compare the point clouds from the optimal design found by the MLMC estimator to the MC reference solution. Additionally, we can also compare point clouds to the optimal solution using the two metric $\mathrm{Dist}_C^{\mathbb{X}}$ and $\mathrm{Dist}_{\mathrm{RMSdev}}^{\mathbb{X}}$. We will now revisit our first benchmark application in Problem 18.

19.2 Problem 18

In this section, we extend the sampling results for Problem 18—as discussed in Section 13.1—to OUU. Additional to the deterministic objective function,

$$f_{\mathrm{det}}(x) = \begin{cases} (x-2)^2 & \text{if } x \leq 3 \\ 2\ln(x-2)+1 & \text{if } x > 3, \end{cases} \tag{19.1}$$

that we want to minimize, we introduce a linear constraint $g : \mathbb{R} \to \mathbb{R}$:

$$g(x) = \frac{4\ln(1.5)}{5}(x-1). \tag{19.2}$$

We employ the same stochastic four-level structure as in Section 13.2

$$\begin{aligned} f_4(x,\theta) &= f_{det}(x) + \theta^3, \\ f_3(x,\theta) &= f_{det}(x) + 1.1\theta^3, \\ f_2(x,\theta) &= f_{det}(x) + \left(\frac{1}{60}x + 1.2\right)\theta^3, \\ f_1(x,\theta) &= f_{det}(x) + \frac{3}{2}\theta^3, \end{aligned} \tag{19.3}$$

where $\theta \sim \mathcal{U}(-0.5, 0.5)$ with the same cost ratio of $\frac{C_i}{Ci-1} = 10$, with $C_4 = 1$, s.t., $C_1 < C_2 < C_3 < C_4$. Compared to the sampling case, where we were restricted at a specific location $x = 1$, it is worth noting the contribution of θ is dependent on x for $f_2(x,\theta)$. As a result, we observe varying correlations across the levels and, consequently, different resource allocations across x. Hence, the resource allocation changes throughout the optimization over x. Therefore, we construct the new MLMC estimators and compute the optimal resource allocation in each optimization step. We present an example on this next.

Example 13 (Variance and resource allocation for MLMC targeting the mean)**.** *We visualize the variance over levels as used for the resource allocation in Eq.* (6.26) *and the resulting resource allocation for the MLMC estimator targeting the mean in Fig. 19.1. We observe how the variance and thus the resource allocation varies over x, due to this previously mentioned dependence. For Problem 18 and the MLMC targeting the mean, these values are computed analytically for this example.*

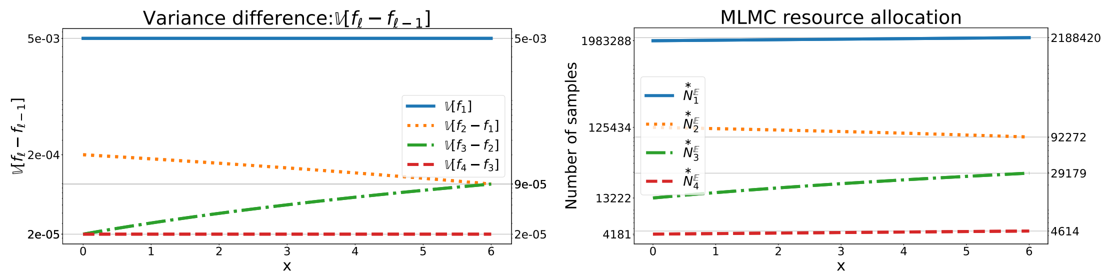


Figure 19.1: The variance of level difference as used in the resource allocation (left) in Eq. (6.26) and the resulting resource allocation (right) for the MLMC estimator targeting the mean for all four levels over x.

To simplify the presentation, we limit our algorithmic choices based on the previous results and confine ourselves to using 20 iterations in conjunction with numerical optimization. We continue to compare the three different methods for approximating the covariance term for scalarization. We are interested in solving the following two optimization problems:

First, we regard the expected value, $\mathcal{R}_0^f$, as reference test case as:

$$\begin{aligned} &\min_x \mathbb{E}[f_4(x,\theta)], \\ &\text{s.t. } f_{det}(x) \geq g(x). \end{aligned} \tag{19.4}$$

Second, we consider our newly developed scalarization estimators, $\mathcal{R}_2^f$, as:

$$\begin{aligned} &\min_x \mathbb{E}[f_4(x,\theta)] + 3\sigma[f_4(x,\theta)], \\ &\text{s.t. } f_{det}(x) \geq g(x), \end{aligned} \tag{19.5}$$

Here, we select $\alpha = 3$, which is a popular value in the field of OUU.

We use SNOWPAC for the optimization process. We initiate the optimization runs from the starting point $x = 0.25$ and carry out the $M^{\mathbb{X}} = 25$ independent runs for each case utilizing the different estimators discussed in this work. We terminate the optimization after 100 iterations. The final designs found in each optimization run are plotted in the subsequent diagrams. Additionally, we track the average computational cost for a single iteration by keeping a record of the resource allocation throughout the optimization process. The MLMC estimators employ the four levels as detailed in Eq. (19.3).

For all following figures, the blue line represents the objective function; the black line shows the constraint. The small figure in the bottom right corner displays the complete function, while the area around the optimal design is magnified. Each marker denotes the final design discovered by the individual run. We display the results when utilizing a standard MC estimator with $N_4 = 1000$ samples as red dots and compare them to the final design found using a MLMC estimator targeting the mean (blue crosses) and, in the second set of results, to a MLMC estimator targeting the scalarization (green triangles). The yellow dot indicates the optimal design.

The results are computed using Dakota, where we use the coupling as described in Chap. 18. We give an example of the input file for Dakota for the scalarization case in the appendix in Lst E.1.

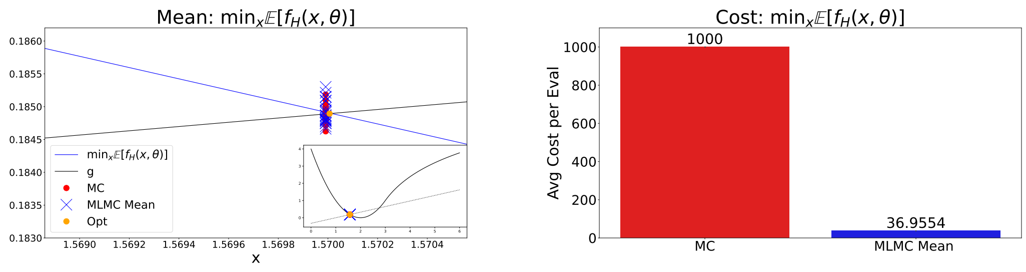
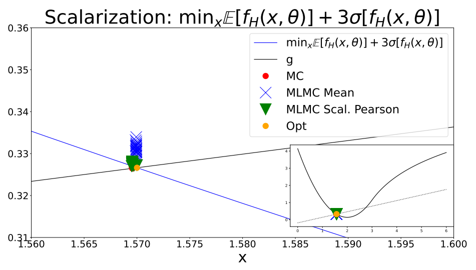


Figure 19.2: Mean: The optimization outcomes for 25 separate runs following 100 iterations when targeting formulation Eq.19.4 for MC (red dots) and MLMC targeting the mean (blue crosses)

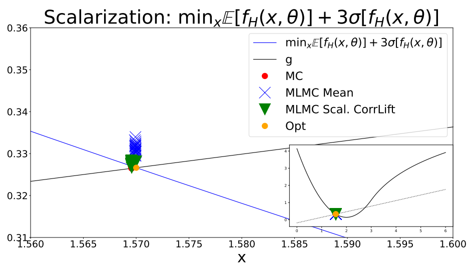
In the first scenario, we target the expectation, where the optimization problem is solved as per Eq. (19.4) aiming for $\epsilon_{\mathbb{E}}^2 \approx$ 2.2321e-6 and present the results in Fig. 19.2.

Here, the final designs obtained by the MLMC mean (depicted by blue crosses) align well with the MC designs (indicated by red dots). Observing the average computational cost per iteration, the considerable advantage of utilizing MLMC methods becomes apparent. In this scenario, we are capable of reducing the computational cost by approximately a factor of 20, given this specific choice of computational cost.

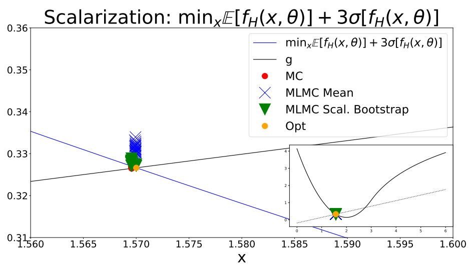
The impact of the MLMC estimator selection becomes evident when we transition to the scalarization case, as outlined in Eq. (19.5), where we apply $\epsilon^2_{\mathbb{E}} \approx 2.2321\text{e-}6$ and $\epsilon^2_{\mathbb{S}} \approx 1.6175\text{e-}5$ for the respective approaches, as shown in Fig. 19.3. The outcomes for different alternatives for approximating the covariance are displayed in the three plots: the Pearson upper bound in Fig. 19.3a, the Correlation Lift in Fig. 19.3b, and the Bootstrap approximation in Fig. 19.3c.



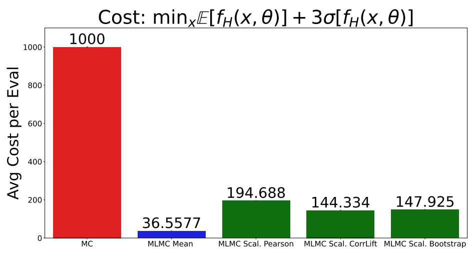
(a) **Pearson** upper bound as described in Section 12.4.1 to bound the covariance term of Eq. (12.27).



(b) **Correlation Lift** approximation as described in Section 12.4.3 to approximate the covariance term of Eq. (12.27).



(c) **Bootstrap** approximation as described in Section 12.4.2 to approximate the covariance term of Eq. (12.27).



(d) Cost average for a single evaluation for the different approaches. The cost are averaged over 25 optimization runs with 100 iterations each.

Figure 19.3: Scalarization: covariance approximation and cost comparison. Optimization results for 25 individual runs after 100 iterations when targeting formulation Eq.19.5 for MC (red dots), MLMC targeting the mean (blue crosses) and MLMC targeting the scalarization (green triangles). In the first three figures, we contrast the different approximation strategies for the covariance term. The fourth figure shows a cost comparison for all three approaches.

As in the sampling investigation in Section 13.1, we observe a much greater variation in the optimal designs identified by the MLMC approach targeting the mean, represented as blue crosses. Consequently, we observe a bias in the distribution of points targeting the mean, which stems from the increased noise introduced by these samples and accounted for in SNOWPAC's optimization process. However, our newly devised scalarization estimators in green match well with the Monte Carlo reference solution in red.

This is also mirrored in the average computational cost for a single evaluation in Fig. 19.3d. We note a very small average evaluation cost for the MLMC approach targeting the mean (blue bar). The estimator is underresolved, as it does not use enough samples, leading to a larger estimator variance. For the three MLMC strategies targeting the scalarization (green bars), we see a cost reduction relative to the reference MC, albeit less than for MLMC targeting the mean. This cost, however, aligns with the variance of the reference solution.

When comparing the costs of the three strategies for approximating the covariance, we find a familiar pattern: the highest cost is associated with the Pearson upper bound, as it is indeed an upper bound and takes a conservative approach; the Correlation Lift approximation and the Bootstrap approximation have slightly lower costs, but we must consider additional computational cost for the Bootstrap approach. Therefore, we find the Correlation Lift to be the most cost-efficient strategy.

Lastly, we assess the approximation quality of the new methods through quantitative metrics displayed in Table 19.1. Using the metrics detailed in Section 19.1, we compute the distance to the final designs relative to the reference MC solution. It is evident that our novel MLMC method, which targets scalarization, is consistently closer to the reference designs than the standard MLMC approach that targets the mean. The smallest value in each column is highlighted in bold. For this scenario, the Pearson upper bound appears to be a viable conservative choice.

Method	$\mathbb{X}$	$\mathrm{Dist}_{\mathrm{C}}^{\mathbb{X}}$	$\mathrm{Dist}_{\sigma}^{\mathbb{X}}$	$\mathrm{Dist}_{\mathrm{RMSdev}}^{\mathbb{X}}$
MLMC Mean	$\mathbb{E}$	4.1683e-3	7.2126e-4	4.3140e-3
MLMC Scal. (Pearson)	$\mathbb{S}$	**5.6635e-4**	1.1207e-4	**5.4988e-4**
MLMC Scal. (CorrLift)	$\mathbb{S}$	7.0536e-4	5.9096e-5	7.4629e-4
MLMC Scal. (Bootstrap)	$\mathbb{S}$	8.0911e-4	**1.8353e-5**	8.5709e-4

Table 19.1: Quantitative comparison of the proximity of the discovered final designs to the MC reference solution. Each row corresponds to a different approach. The last three columns indicate a distinct metric, while the second column specifying the target of the estimator.

To summarize the findings from this section, we have demonstrated the efficiency of the newly developed estimators for OUU. It is important to tailor the MLMC estimator according to the specific formulation of the optimization problem at hand. We emphasized the importance of accurately approximating the covariance for the scalarization scenario, which motivated the development of these new estimators. In the forthcoming section, we will tackle a more complex optimization problem, namely, the constrained Rosenbrock function, where we will construct a three-level test case.

19.3 Rosenbrock problem

As our second case, we utilize the constrained 2-D Rosenbrock optimization problem as outlined in [289], a popular benchmark within the optimization field. In its deterministic format, it is represented as follows:

$$
\begin{aligned}
\min_{x_1,x_2} & f(x_1,x_2) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2, \\
\text{s.t. } & c_1 : (x_1 - 1)^3 + 1 - x_2 \leq 0, \\
& c_2 : (x_1 + x_2) - 2 \leq 0.
\end{aligned}
\tag{19.6}
$$

The problem is depicted for $x_1 \in [-1.5, 1.5]$ and $x_2 \in [-0.5, 2.5]$ in Fig. 19.4. The unconstrained problem has a unique global minimum at $(1, 1)$, whereas this constrained problem possesses a local minimum at $(0, 0)$, along with a global minimum at $(1, 1)$. Small gradients render it difficult for optimization algorithms to identify the global minimum.

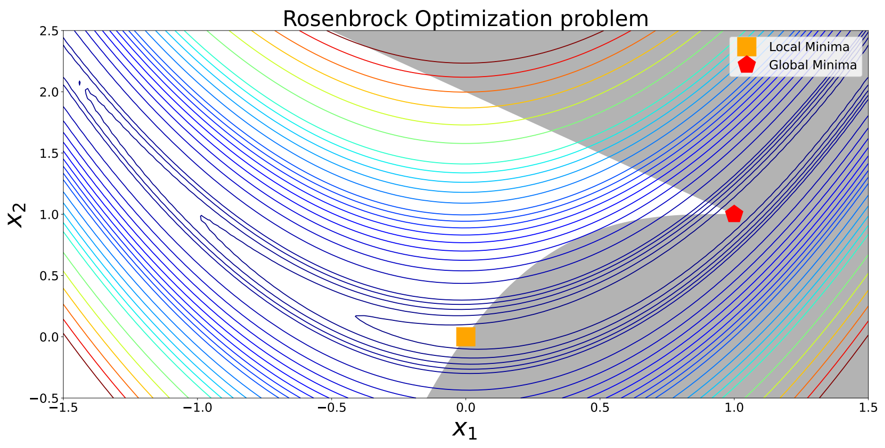


Figure 19.4: Graphical representation of the Rosenbrock function optimization problem. The contour lines of the objective function are plotted along with the two grey constraints. The infeasible region is marked in grey. The local and global optima are represented as an orange square and red pentagon, respectively.

To transform the deterministic problem into a stochastic one with multiple levels, we employ the Ishigami function, adapted from the work in [264]. The specifics of the three functions along with their respective mean and sigma are tabulated in Table 19.2. Here, $\{\theta_i\}_{i=1}^3 \sim \mathcal{U}(-\pi, \pi)$ follow a uniform distribution, with $a = 5$ and $b = 0.1$.

We combine the Ishigami function with the Rosenbrock objective function $f(x_1, x_2)$ yield three levels: $\{f(x_1, x_2) + \beta I^{(i)}(\theta_1, \theta_2, \theta_3)\}_{i=1}^3$. Moreover, to standardize the stochastic influence of the Ishigami function in relation to the deterministic Rosenbrock function, we introduce a scaling factor $\beta = \sqrt{1\text{e-}4}$. We also assume the cost ratio among the distinct levels to be $\frac{C_i}{C_{i-1}} = 10, i = 2, 3$, with $C_3 = 1$, so that $C_1 < C_2 < C_3$.

Level function	$\mu_1^{(i)}$	$\sigma^{(i)}$
$I^{(3)}(\theta_1,\theta_2,\theta_3) = \sin(\theta_1) + a\sin(\theta_2)^2 + b\theta_3^4$	2.5	3.2931
$I^{(2)}(\theta_1,\theta_2,\theta_3) = \sin(\theta_1) + 0.85a\sin(\theta_2)^2 + b\theta_3^4\sin(\theta_1)$	2.125	3.1595
$I^{(1)}(\theta_1,\theta_2,\theta_3) = \sin(\theta_1) + 0.6a\sin(\theta_2)^2 + 9b\theta_3^2\sin(\theta_1)$	1.5	3.5308

Table 19.2: Three levels of the Ishigami function $\{I^{(i)}\}_{i=1}^3$ and their corresponding mean $\mu_1^{(i)}$ and standard deviation $\sigma^{(i)}$.

Consequently, this results in an optimization problem analogous to the earlier described Problem 18, where we examine the formulation $\mathcal{R}_2^f$:[1]

$$\begin{aligned}
&\min_{x_1,x_2} \mathcal{R}_2^f[f+I^{(3)}] - \mu_1^{(3)} - 3\sigma_3 = \mathbb{E}[f+I^{(3)}] + 3\sigma[f+I^{(3)}] - \mu_1^{(3)} - 3\sigma^{(3)},\\
&\text{s.t. } c_1 : (x_1-1)^3 + 1 - x_2 \le 0,\\
&\qquad c_2 : (x_1+x_2) - 2 \le 0.
\end{aligned} \tag{19.7}$$

We highlight that we subtract the mean value $\mu_1^{(3)}$ and the standard deviation $\sigma^{(3)}$ to align the solution with the deterministic scenario. As a result, the local and global optima remain the same as those in the deterministic case.

We follow a similar methodology to our previous study: starting from the initial point $(x_1, x_2) = (0.25, 1.5)$, we conduct $M^{\mathbb{X}} = 25$ separate optimization runs for each of the different methods, calculating the MLMC estimator. We look at two different formulations for the MLMC resource allocation. In the first case, we fix the variance of the estimator, minimizing the computational cost, as we have described throughout this work. In the second case, we fix the cost of the estimator and minimize its variance, as described in Rem. 1.

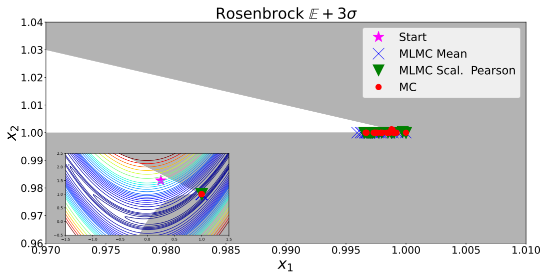
19.3.1 Variance constraint

The targets $\epsilon_{\mathbb{E}}^2 \approx$ 1.0849e-5 and $\epsilon_{\mathbb{S}}^2 \approx$ 8.8951e-5 are again determined by the reference variance of a MC estimator with $N_3 = 1000$ samples. To facilitate a fair comparison, we limit all optimization runs to 250 iterations each and plot the final design found for all the different runs. We also contrast the average computational cost per single iteration. In order to decrease the volume of results, we fix the resource allocation to use 20 iterations and a numerical optimization. This decision is based on past results that showed the best performance. Finally, we once more compare the different strategies of covariance approximation.

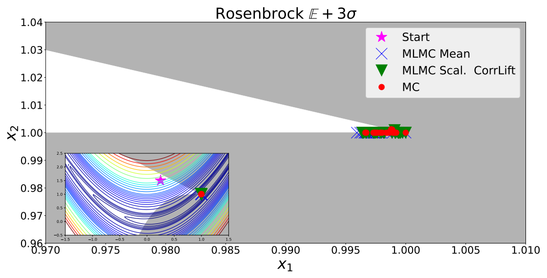
The optimization results from 25 distinct runs are presented in Fig. 19.5. We proceed with a comparative analysis of the three methods of approximating the covariance function in Figs. 19.5a- 19.5c. As before, the final optimal designs derived from the MC approach are symbolized by red circles. The optimal designs acquired via the standard

[1]Please note that we are excluding the formulation $\mathcal{R}_0^f$ for this case as the MLMC estimator is not part of our contribution.

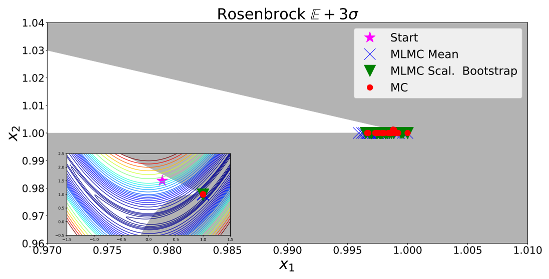
MLMC approach, which targets the mean, are denoted by blue crosses. The MLMC estimator targeting scalarization, which is our novel method, is represented by green triangles in the designs.



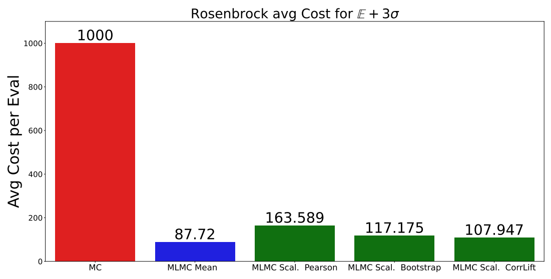
(a) We use the Pearson upper bound as described in Section 12.4.1 to bound the covariance term of Eq. (12.27).



(b) We use the Correlation Lift approximation as described in Section 12.4.3 to approximate the covariance term of Eq. (12.27).



(c) We use the Bootstrap approximation as described in Section 12.4.2 to approximate the covariance term of Eq. (12.27).



(d) Cost average for a single evaluation for the different approaches. The cost are averaged over 25 optimization runs with 250 iterations each.

Figure 19.5: Optimization results for 25 individual runs after 250 iterations for MC (red dots), MLMC targeting the mean (blue crosses) and MLMC targeting the scalarization (green triangles). In the first three figures, we contrast the different approximation strategies for the covariance term. The fourth figure shows a cost comparison for all three approaches.

The results are computed using Dakota, where we use the coupling as described in Section 18. We give an example of the input file for Dakota for the scalarization case in the appendix in Lst. E.2.

The results show a strong alignment between our newly developed MLMC estimators and the MC reference solution. Additionally, the standard MLMC method demonstrates superior performance compared to the results previously displayed. However, it is noteworthy that their final design set reveals increased variance. The cause is apparent when we examine the costs on the right side: the standard MLMC estimator recurrently underresolves the estimators, yielding higher variance for the estimator and more noise in SNOWPAC. As discussed in Part IV, the intensity of the noise is a crucial determinant of SNOWPAC's convergence.

Regarding the cost of approximating the covariance, we observe a scenario similar to the prior example: the Pearson approximation tends to be overly conservative, thereby incurring unnecessary computational cost. The Bootstrap and Correlation Lift approximations yield similar, reduced costs, but if we include the additional computational expenses of bootstrapping, the Correlation Lift approximation seems more advantageous.

For a more detailed numerical analysis, we again employ various metrics (Section 19.1) to calculate the difference between the MLMC methods' optimal designs and the MC reference solution, as demonstrated in Table 19.3. All metrics indicate that our newly introduced method consistently approximates the reference solution more closely across all covariance approximations. The lowest value in each column, signifying the optimal outcome, is highlighted in bold.

Method	$\mathbb{X}$	$\mathrm{Dist}_{\mathrm{C}}^{\mathbb{X}}$	$\mathrm{Dist}_{\sigma}^{\mathbb{X}}$	$\mathrm{Dist}_{\mathrm{RMSdev}}^{\mathbb{X}}$
MLMC Mean	$\mathbb{E}$	1.3519e-3	[1.9332e-4, 3.1944e-4]	1.3106e-3
MLMC Scal. (Pearson)	$\mathbb{S}$	1.2922e-3	[3.1977e-4, 2.7096e-4]	1.2249e-3
MLMC Scal. (CorrLift)	$\mathbb{S}$	1.2686e-3	[3.1931e-4, **1.3793e-4**]	1.1886e-3
MLMC Scal. (Bootstrap)	$\mathbb{S}$	**1.1108e-3**	[**9.8059e-5**, 3.2164e-4]	**9.7472e-4**

Table 19.3: Quantitative comparison of the proximity of the discovered final designs to the MC reference solution. Each row corresponds to a different approach. The last three columns indicate a distinct metric, while the second column specifies the target of the estimator.

Similar to Problem 18 from Section 19.2, we see once more that it is crucial to align the MLMC allocation target with the corresponding configuration of the optimization problem. Fine-tuning the estimation algorithm to the SoI ensures reliably and efficiently achieving the desired accuracy. We also again point out the possible cost reduction when employing MLMC methods compared to standard MC.

19.3.2 Cost constraint

In the following section, we ask ourselves the questions: how does the new MLMC estimator perform when we fix the computational cost of the MLMC estimator to fit the cost of the MC reference solution? How does that subsequently impact the noise and trust-region radius and, thus, convergence in SNOWPAC? Since the MLMC estimator is supposed to reduce the variance, does this translate to a higher accuracy in the optimization and faster convergence?

For this setting, we take the same setup as before, with a change in the resource allocation: instead of fixing the variance and minimizing the total computational cost of the estimator, we now fix the computational cost to $C_T = 1000$ and minimize the estimator variance. Thus, we follow the formulation for the resource allocation as described in Rem. 1. As reference, we use a MC estimator using 1000 samples. We again do 25 optimization runs for each approach, which we average over and compare the new MLMC estimator targeting the scalarization to the MC estimator. We limit all optimization runs to 250 iterations each and plot the final design found for all the different runs. Then, we contrast the variance of the estimators over the iterations and further compare the resulting noise $\tilde{\varepsilon}^f$ in SNOWPAC and the eventual trust-region radius ρ. Finally, we quantitatively compare which approach is closer to the optimal solution $(1, 1)$ after 250 iterations.

In order to decrease the volume of results, we again fix the resource allocation to use 20 iterations and a numerical optimization. Furthermore, we employ the Correlation Lift for the approximation of the covariance term. The results are again computed by Dakota, where we use the coupling as described in Section 18. The input is based Lst. E.2, only changing the `convergence_tolerance_target` to `cost_constraint` as given in the commented section in that listing.

The optimization results from 25 distinct runs are presented in Fig. 19.6 on the left. We clearly see that the MLMC solutions on average are closer to the optimum compared to the MC reference solution. When we regard the variance of the sample estimators, we see the reason: on the one hand, the MC estimator variance fluctuates around its exact variance of $\epsilon_{\mathbb{S}}^2 \approx 8.8951\text{e-}5$ we used in the previous section as variance constraint. On the other hand, the MLMC estimator is able to reduce the variance of its estimator by optimizing the resource allocation, while having the same computational cost as MC.

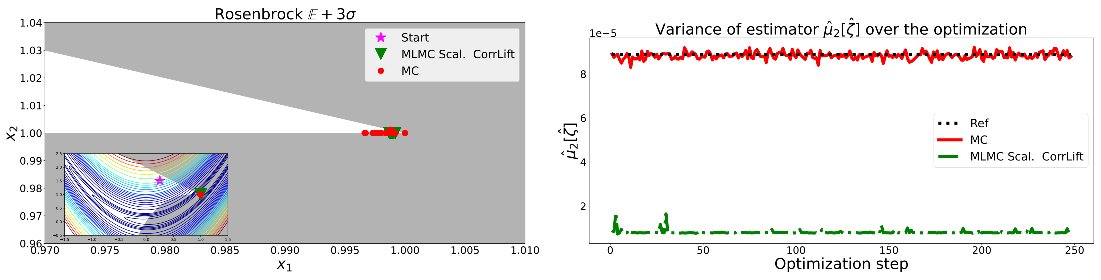


Figure 19.6: Left: Optimization results for 25 individual runs after 250 iterations for MC (red dots) and MLMC targeting the scalarization (green triangles) for cost constraint in the resource allocation (left). Right: Variance of the MC (red, solid) and MLMC targeting the scalarization (green, dashed dot) estimator compared to the exact MC variance using 1000 samples (black, dotted).

How does this translate to SNOWPAC? We plot the noise value of $\tilde{\varepsilon}_k^f$ on the left and the trust-region radius ρ_k over the number of iterations on the right of Fig. 19.7. Note here, that $\tilde{\varepsilon}_k^f$ is already the value after GP smoothening. We clearly see the smoothening effect, where the GP is able to reduce the noise, when we compare it to the variance of the sample estimator from the previous figure. Moreover, MC and MLMC have similar values at the start. However, we also note the reduced variance of the MLMC estimator for increasing iterations resulting in a lower noise. The lower noise is the result of two factors: first, the reduced variance from the MLMC estimator; second, a smaller noise results in a smaller trust-region radius. We observe this in the right plot, where the trust-region radius of MLMC is visibly smaller than MC over the optimization. The smaller trust region results in more evaluations in closer vicinity and improves the GP surrogate faster. Ultimately, this results in faster convergence of the method in this case, since SNOWPAC converges based on a bound on the trust-region radius.

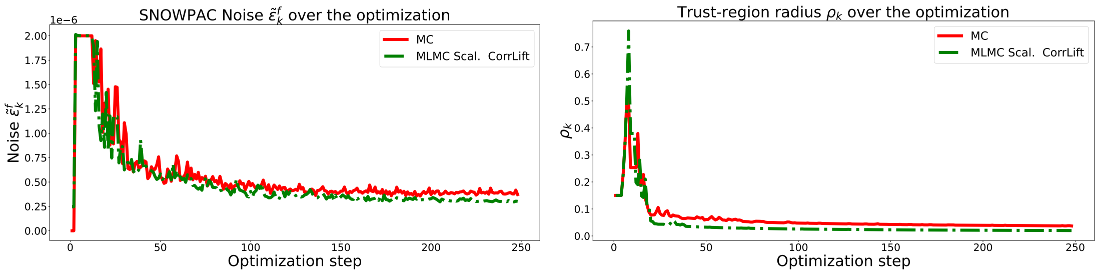


Figure 19.7: Left: Noise value $\tilde{\varepsilon}_k^f$ of the current best design at iteration k for MC (red, solid) and MLMC targeting the scalarization (green, dashed dot). Right: Trust-region radius ρ_k at iteration k for MC (red, solid) and MLMC targeting the scalarization (green, dashed dot).

Quantitatively, we compare both the MC and the MLMC approach targeting the scalarization to the optimal solution $(1, 1)$ in Table 19.4. Since we compare to a single point, we show the difference of the cluster centers, $\mathrm{Dist}_{\mathrm{C}}$, and the root-mean-square deviation, $\mathrm{Dist}^{\mathbb{X}}_{\mathrm{RMSdev}}$, to $(1, 1)$. For both metrics, we also see the quantitative improvement of using the MLMC estimator, resulting in a more accurate solution compared to MC for the same computational cost.

Method	$\mathbb{X}$	$\mathrm{Dist}^{\mathbb{X}}_{\mathrm{C}}$	$\mathrm{Dist}^{\mathbb{X}}_{\mathrm{RMSdev}}$
MC	$\mathbb{S}$	1.7912e-3	2.0122e-3
MLMC Scal.	$\mathbb{S}$	**1.0766e-3**	**1.0840e-3**

Table 19.4: Quantitative comparison of the proximity of the discovered final designs to the exact solution $[1, 1]$. We compare the MC and new MLMC approach targeting the scalarization. The last two columns indicate a distinct metric, while the second column specifies the target of the estimator.

Note, that these results are from specific test conditions, where the variance of the estimator is constant over the domain. Otherwise, the path of optimization plays a role in the variance of the estimator, since the variance changes over the design space. Nevertheless, in this second case, we see that, given the same computational cost, the MLMC estimator is able to find a better local optimum faster. In the end, this results in faster convergence of the algorithm in this case.

To sum it up for both test cases, we observe enhanced optimization performance when implementing our newly devised estimators, even in the face of more demanding test conditions. Both cases combined show the potential of the MLMC method when a hierarchy of levels is available.

19.4 Optimization under uncertainty for wake steering

In this final result section, we are interested in formulating and solving an OUU problem for finding the optimal wake steering strategy of a wind power plant. We design a multilevel case, employing our newly devloped MLMC estimators with SNOWPAC. First, we discuss general wind power plant design in Section 19.4.1 before we talk about wake steering in Section 19.4.2. Finally, we are going show results for two wake steering formulations, for the expected total power production and the scalarized total power production in Section 19.4.4.

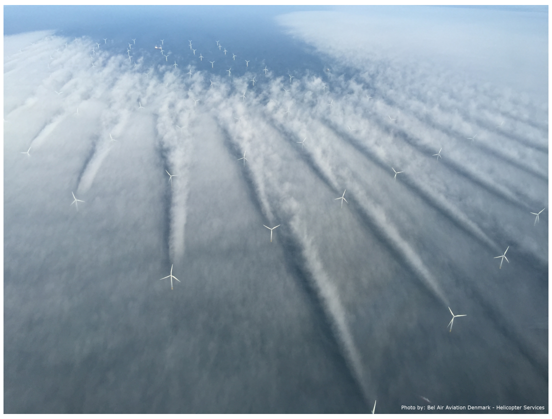
19.4.1 Wind power plant design

Wind power plants, or wind farms, are critical components of the global renewable energy infrastructure. They represent a clean, sustainable solution for generating electricity that reduces our reliance on fossil fuels and helps to mitigate the effects of climate change. They usually comprise several wind turbines ranging from a few to hundreds, depending on the scale of the farm. These turbines are usually installed in areas with high wind speeds, such as coastal regions, open plains, or on hills and ridges. Wind farms can also be offshore, where the wind is typically stronger and more consistent.

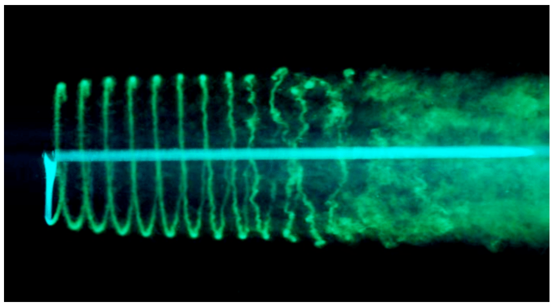
The energy production process in a wind power plant begins when the wind turns the turbine blades. This rotational energy is then converted into electrical energy by a generator located within the turbine. Wind power is a variable source of energy, as its generation depends on wind speed and direction. Despite the variability, wind power provides numerous benefits: it is a renewable resource, meaning it will not deplete over time like fossil fuels. Furthermore, wind power plants do not produce greenhouse gas emissions while operating, contributing significantly to the reduction of carbon footprints [11].

A lot of planning goes into finding the optimal design of a wind power plant [83]. We highlight three important factors. First, we need to find an optimal location, which includes many considerations such as wind conditions (constant or high variations), the connection to the power grid or the impact on the surroundings. Second, we need to decide on the arrangement of turbines in the field. Here, we have to consider wake effects. Wake describes the disturbed flow of air behind a wind turbine which interferes with turbines downwind and hence can decrease the overall power production. Third, we have to think about the turbine types we want to use considering recent developments for horizontal but also vertical wind turbines [234].

We show one of the most popular examples for real wake effects in Fig. 19.8a. This shows the Horns Rev II wind form located in the North Sea, 30 km west of Blåvandshuk on the Danish west coast. The farm consists of 91 wind farms with a total capacity of 209 MW. The turbine towers have a height of 68 metres with a turbine blade diameter of 93 metres. The wind farm was opened in 2009 and is operated by Ørsted [340]. Due to the foggy weather conditions in the picture, the wake of the individual turbines and how it interferes with downwind turbines is clearly visible. An example of an experimentally created turbine wake by a one-bladed rotor is given in Fig. 19.8b. We observe the helical vortices, which break down into a turbulent flow. The wake is created in a dye visualization in water [201].



(a) Horns Rev II wind farm off the coast of Denmark The wake effects are visible due to low hanging fog. Photo by: Bel Air Aviation A/S [12]. January 26th, 2016. Used with permission.



(b) Dye visualization in water of wake effect behind one-bladed rotor [201]. Licensed under CC BY 3.0.

Figure 19.8: Example of a wind farm and experimental turbine wake.

19.4.2 Wake steering

Once a wind plant has been installed, its performance can be further improved by implementing control techniques. Wake steering is one such effective control technique, where turbines are purposefully yawed off the path of oncoming wind, instead of aligning perpendicularly [266, 267]. This causes the turbines to function as flow regulators, directing the wake away from downstream turbines and drawing more momentum from higher elevations. However, designing effective wake-steering strategies can be challenging due to variables such as the direction and intensity of incoming wind, turbulence levels, and other factors. In addition, inaccuracies in wind sensors and yaw control systems add further complexity in creating optimal wake steering strategies. The turbulent flow within the turbine's wake, characterized by high-vorticity regions interacting with other wakes and downstream rotors, results in sensor readings with considerable uncertainty. This makes it difficult to accurately determine each turbine's yaw angle.

Control research often relies on low-fidelity, but cost-effective tools that simplify governing equations through linearization and layering of analytical wake shapes. Floris [243] serves as a control-centric model for wake-steering simulation, offering a variety of wake parametrizations, some of which consider the yaw-induced wake deflection and curl. Wake deflection involves the shift in the wake's path due to the yaw motion, which is the turbine's rotation around a vertical axis. Meanwhile, wake curl refers to the swirling or torsional motion within the wake, again, as a result of the yaw motion. These components play a vital role in creating efficient wake-steering strategies, aimed at optimizing a wind farm's power output by controlling the influence of the wakes from upstream turbines on those downstream.

One SoI of wake-steering scenarios is the average total power (ATP). It is given as the expectation of the wind farm power output with T turbines for a given set of yaw angles with respect to the site-specific joint distribution of wind speed and direction:

$$\begin{aligned} \mathrm{ATP}\left(\boldsymbol{\gamma}, u, \phi\right) &:= \mathbb{E}\left[f_{\text{power}}\left(\boldsymbol{\gamma}, u, \phi\right)\right] \\ &= \mathbb{E}\left[\sum_{i=1}^{T} f_{\text{power}}\left(\gamma_i, u, \phi\right)\right] \end{aligned} \tag{19.8}$$

where $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_T]^T$ represents the vector of yaw angles, which we control, while the inflow wind speed u and inflow wind angle ϕ are random variables.

Another SoI is the variation in the total power (VTP). Here, we consider the standard deviation of the plant's power output for the given set of yaw angles with respect to the site-specific joint distribution of wind speed and direction,

$$\begin{aligned} \mathrm{VTP}\left(\boldsymbol{\gamma}, u, \phi\right) &:= \sigma\left[f_{\text{power}}\left(\boldsymbol{\gamma}, u, \phi\right)\right] \\ &= \sqrt{\mathbb{V}\left[\sum_{i=1}^{T} f_{\text{power}}\left(\gamma_i, u, \phi\right)\right]}. \end{aligned} \tag{19.9}$$

We then consider two OUU problems:

1. We optimize for the maximal expected ATP:

$$\begin{aligned} &\max_{\boldsymbol{\gamma}} \mathcal{R}_0^f(\boldsymbol{\gamma}) := \max_{\boldsymbol{\gamma}} \mathrm{ATP}\left(\boldsymbol{\gamma}, u, \phi\right) \\ &\text{s.t.} \quad |\boldsymbol{\gamma}_i| \leq 30^\circ \quad \text{for} \quad i = 1, \ldots, T. \end{aligned} \tag{19.10}$$

2. We maximize the ATP subject to a set of constraints on yaw angles, while minimizing its variability, the VTP:

$$\begin{aligned} &\max_{\boldsymbol{\gamma}} \mathcal{R}_2^f(\boldsymbol{\gamma}) := \max_{\boldsymbol{\gamma}} \mathrm{ATP}\left(\boldsymbol{\gamma}, u, \phi\right) - \alpha \mathrm{VTP}\left(\boldsymbol{\gamma}, u, \phi\right), \\ &\text{s.t.} \quad |\boldsymbol{\gamma}_i| \leq 30^\circ \quad \text{for} \quad i = 1, \ldots, T. \end{aligned} \tag{19.11}$$

   where we control the weight of the VTP through α.

19.4.3 Wind farm problem setup

We consider a wake steering problem for a wind farm with $T = 9$ turbines. The turbine towers are 90 meters in height and the turbine rotors have a diameter of $D = 126$ meters and imitate the NREL 5MW turbine [171]. We have three clusters of turbines with three turbines each. In each cluster, the turbines are offset to each other by $3.5D$ meters in x-direction, and about $0.5D$ meters in y-direction. Hence, we get wake effects in each cluster of turbines. The distance between clusters is also set to about $3.5D$ meters such

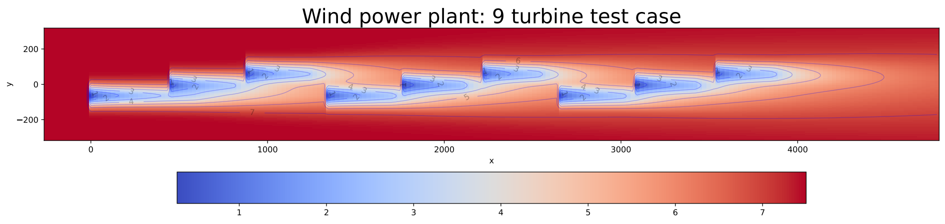


Figure 19.9: Wind farm setup with nine turbines facing the inflowing wind.

that we also expect wake effects between cluster. We present the setup in Fig. 19.9 with a nominal wind speed of $u = 7.5\frac{m}{s}$ and a wind angle of $\phi = 270°$ (flow from left to right).

For the stochastic setting, we introduce randomness on the wind speed and wind angle. We use $u \sim \mathcal{N}(7.5\frac{m}{s}, 1\frac{m}{s})$ and $\phi \sim \mathcal{N}(270°, 5°)$ as new inflow conditions based on the setting in [267]. In addition, we define a three-level setting for the multilevel case in Table 19.5. For the finest level 3, we use 40 points for the turbine discretization and a Gaussian velocity and deflection model for the wake effects (see [244] and references therein for details about wake models). For the medium level 2, we reduce the turbine discretization to a single point, but employ the same wake model. Finally, for the coarse level 1, we also employ a single discretization point but use no wake model. This results in the given relative computational cost structure for a single evaluation of the model on each level as given in the last column.

Level ℓ	Turbine grid points	Wake model	Relative cost C_ℓ
3	40	GCH	1
2	1	GCH	0.12
1	1	No wake	0.002

Table 19.5: Three-level design for MLMC wake steering scenario.

For these results, we cannot fix a target variance accuracy for the MLMC resource allocation to compare to the MC results, since this would require to know the variance of the MC estimator in each optimization step. This is neither analytically available (as it was for Problem 18) nor numerically computable (as it was for Rosenbrock). Thus, we use the resource allocation formulation targeting a certain cost, as described in Rem. 1 and as we presented for the Rosenbrock problem in Section 19.3.2. We set the target cost $C_T = C_3N_3$ with $N_3 = 1000$ samples on the finest level, where the resource allocation optimizes for the variance of the estimator. An example for the Dakota input file of such a setting is given in the appendix in Lst. E.3.

19.4.4 Results

We first look at the optimization results for the maximal expected ATP from Eq. (19.10). We compare the MC estimator with the MLMC estimator targeting the mean, both targeting the same computational cost. We note here that these results not only show our third contribution of coupling MLMC with SNOWPAC, but also the performance of SNOWPAC alone for a complex, high-dimensional black-box problem for the single-level MC case.

In Fig. 19.10, we show the development of the optimal design for each of the two cases. We observe that both approaches result in very similar optimal designs for the yaw angles, albeit using different estimators. For the MLMC case, we see a lot of movement in the design in the first 15 iterations; then the solution quickly seems to converge. The MC estimator seems to take a little longer, with another visible change after about 20 iterations. Given that both approaches seem to be converged afterwards, we manually stopped the MLMC approach at that point after 58 optimization steps.

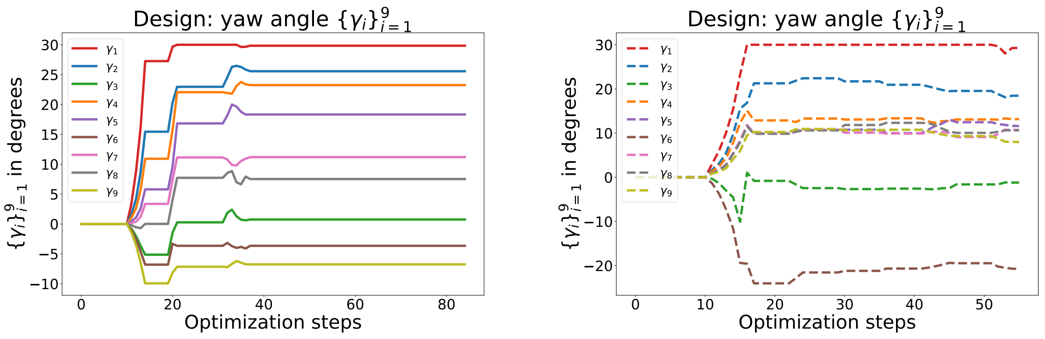


Figure 19.10: Optimization results for the optimal design of yaw angles $\{\gamma_i\}_{i=1}^9$ for Eq. 19.10. We compare optimization results for standard MC (left) and MLMC targeting the mean (right).

This impression is consolidated when looking at the objective value of both approaches over the number of iterations. We visualize this in Fig. 19.11. It seems that the optimizer is able to converge quickly to a local maximum for both approaches. This behavior strengthens the result we received in our benchmark tests for SNOWPAC. Based on the initial movement of yaw angles after 15 iterations, both objectives also increase quickly. We note another jump in the objective for the MC case after the previously mentioned 20 iterations, after which both approaches seem to converge. Additionally, we note that the MC solutions objective seems to be larger than the MLMC objective in the end.

However, we have to be aware that the accuracy of the presented objective value is limited by the accuracy of the employed estimators. Hence, to find the exact objective value, we take the final design found by each approach and evaluate them numerically using MC sampling on level 3 with $N_3 = 10000$ samples over the random variables of u and wind angle ϕ. From these sampling results, we can compute the ATP, the VTP and the objective. We plot the resulting flow field in Fig. 19.12, where we put the values of interest in the title of each subfigure. We compare optimization results for standard MC (second from top), MLMC targeting the mean (third from top) to the inital setting without any wake steering (top).

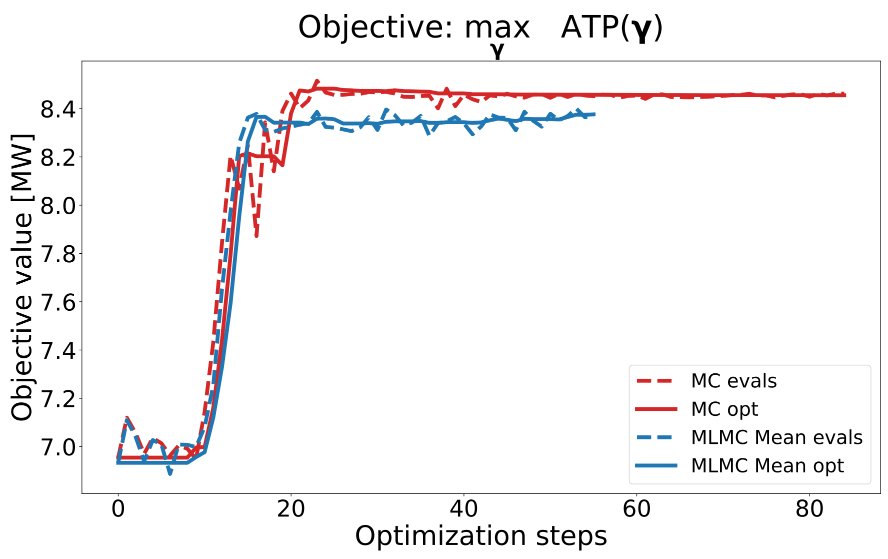


Figure 19.11: Optimization results for the optimal design of yaw angles $\{\gamma_i\}_{i=1}^{9}$ for Eq. 19.10. We compare optimization results for standard MC (red) and MLMC targeting the mean (blue). The optimal path is plotted in solid while evaluations are plotted as dashed line.

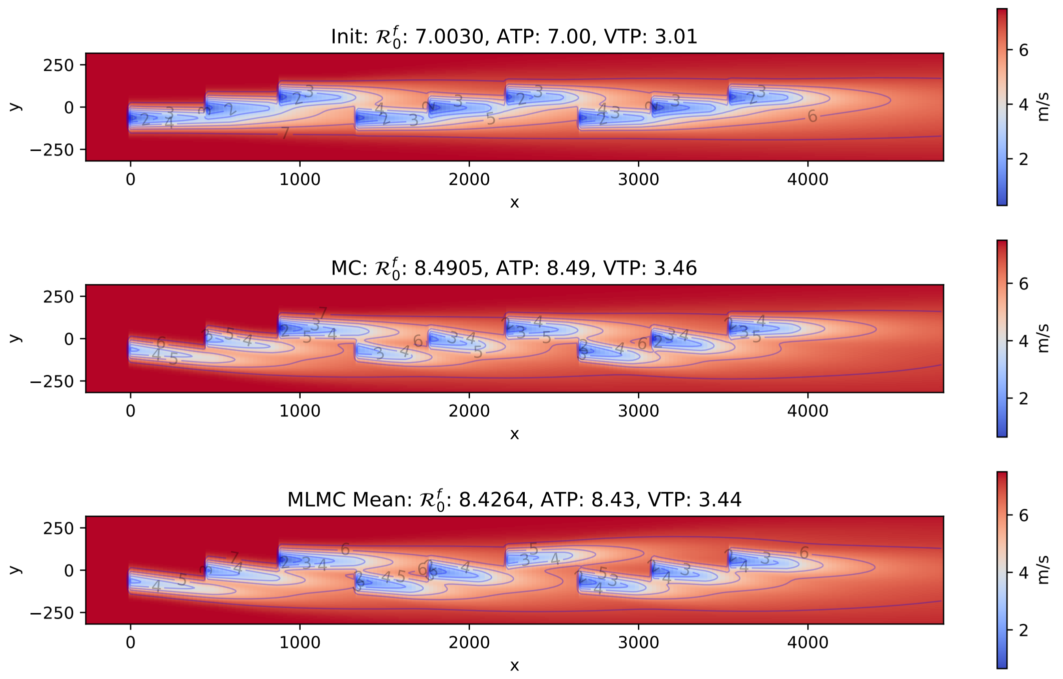


Figure 19.12: Flowfield for different optimal designs found for the different approaches for Eq. 19.10. We compare optimization results for standard MC (center), MLMC targeting the mean (bottom) to the initial setting without any wake steering (top). The values of interest for the objective $\mathcal{R}_0^f$, ATP and VTP for each approach are given in the title of each plot in MW.

In this figure, we first observe that both approaches clearly try to minimize wake effects by turning the turbines such that the inference with downwind turbines is reduced. Looking at the objective value in the title of the results, we see that both approaches find almost a similar local maximum value. Though not relevant for this optimization scenario, we note the VATP value in the plot titles and also see almost no difference there. Thus, we summarize that both approaches are valid for a complex, high-dimensional problem, such as Eq. (19.10) with SNOWPAC finding an optimal result quickly. Finally, we note that we are able to improve the total power production of this fictional wind power plant by approximately 20% by moving the turbines.

Next, we present the results for the scalarization case of Eq. 19.11, where we use $\alpha = 3$. Here, we combine our contribution of SNOWPAC with the contribution of the new MLMC estimator for the scalarization. We compare this contribution to the standard MC and MLMC approaches.

In Fig. 19.13, we show the development of the optimal design for each of the three cases. We first note that the MLMC approach targeting the mean has crashed after about 65 optimization steps. On the one hand, we observe see that the MC solution (top left) and the MLMC approach targeting the scalarization (bottom) tend to the boundary of the allowed yaw angles. We also see a similarity in the design for both approaches. On the other hand, we see the MLMC approach targeting the mean (top right) yields a design with less yaw angle movement.

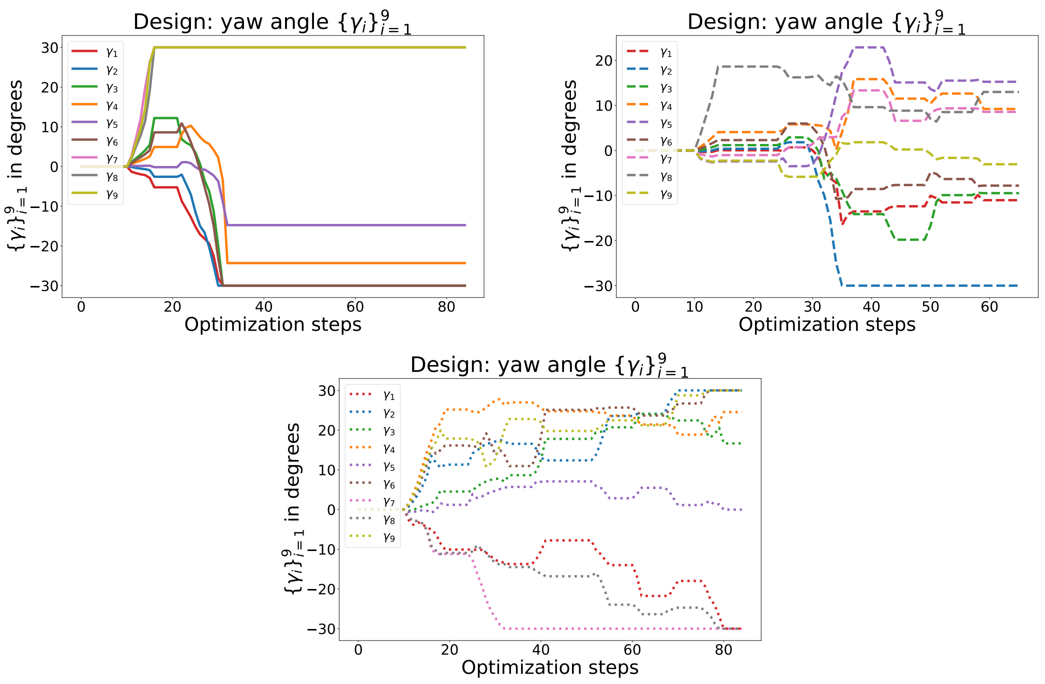


Figure 19.13: Optimization results for the optimal design of yaw angles $\{\gamma_i\}_{i=1}^9$ for Eq. 19.11. We compare optimization results for standard MC (top left), MLMC targeting the mean (top right) and MLMC targeting the scalarization (bottom).

As a result of this movement of the design, we see the development of the objective in Fig. 19.14. Here, we plot the values that SNOWPAC uses for the optimization, which are estimates by themselves. We see the largest value in the objective for the MLMC approach targeting the scalarization in green, whereas the MC solution in red results in a smaller optimum. The MLMC approach targeting the mean finds the smallest objective value, fluctuates much more and does not show a steady increase in the objective.

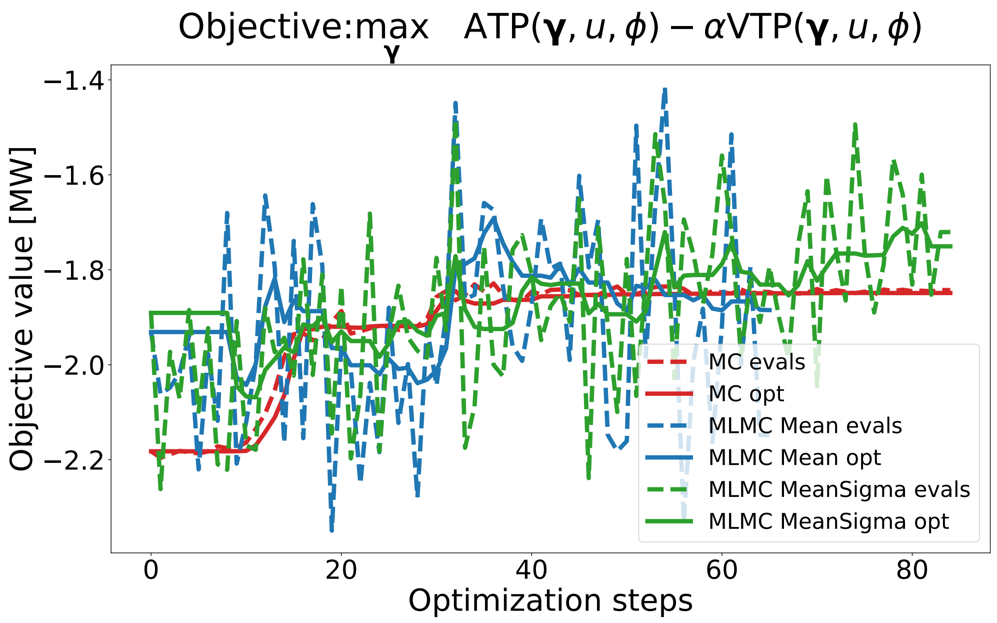


Figure 19.14: Optimization results for the optimal design of yaw angles $\{\gamma_i\}_{i=1}^{9}$ for Eq. 19.11. We compare optimization results for standard MC (red), MLMC targeting the mean (blue) and MLMC targeting the scalarization (green). The optimal path is plotted in solid while evaluations are plotted as dashed line.

Finally, we want to quantify if this translates to an improved result. Hence, we quantify the optimal results that each method found exactly. We again employ the final design found by each approach and evaluate them numerically using MC sampling on level 3 with $N_3 = 10000$ samples over the random variables of u and a wind angle ϕ. We plot the resulting flow field in Fig. 19.15, where we put the computed objective, ATP, and the VTP in the title of each plot. We compare optimization results for standard MC (second from top), MLMC targeting the mean (third from top) and MLMC targeting the scalarization (bottom) to the initial setting without any wake steering (top).

First of all, we observe that all approaches clearly try to steer the turbines to reduce wake effects. We further note that objectives $\mathcal{R}_2^f$ for all approaches are negative; this means that the standard deviation term is dominating the optimization. Regarding the objective values, we see that MLMC targeting the scalarization (bottom) results in the largest objective value. We also observe that our approach, MLMC targeting the scalarization, results in the smallest value for VTP out of all approaches. This comes

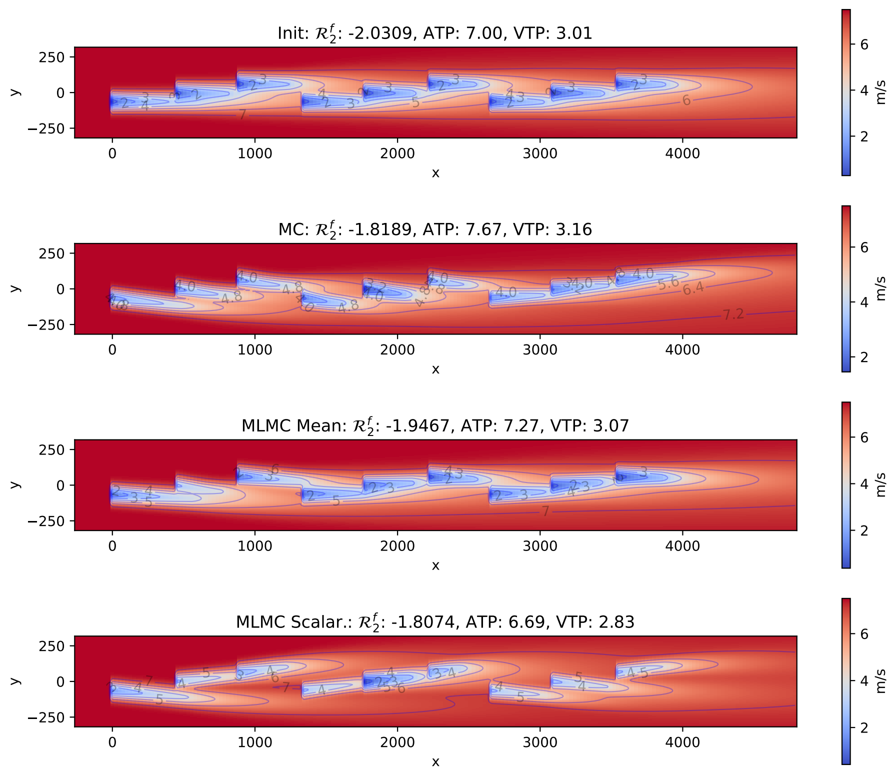


Figure 19.15: Flowfield for different optimal designs found for the different approaches for Eq. 19.11. We compare optimization results for standard MC (second from top), MLMC targeting the mean (third from top) and MLMC targeting the scalarization (bottom) to the initial setting without any wake steering (top). The values of interest for the objective $\mathcal{R}_2^f$, ATP and VTP for each approach are given in the title of each plot in MW.

with the trade-off of a smaller ATP value, which we expect from a robust optimization scenario. The largest ATP value is found by MC, but its overall objective value $\mathcal{R}_2^f$ is smaller compared to MLMC targeting the scalarization. MLMC targeting the mean has neither the largest objective nor does it find the best result for ATP or VTP.

Regarding the resulting power output, we point out that we actually decrease the average total power production of this wind power plant by approximately 5% by moving the turbines. This is a result of our more conservative optimization formulation. However, at the same time, we also decrease the variation in the power production by almost 6%. This is a reasonable formulation in practice if the total power production is not the limiting factor, whereas power spikes should be avoided.

We conclude that our new approach is able to improve the results compared to standard MC, but also specifically compared to the classical MLMC approach, which targets the mean estimator. By adapting the estimator target to the OUU formulation, we are able to achieve better results, since the formulation itself is consistent with the problem.

Lastly, we again see the capabilities of SNOWPAC, being able to converge to a (local) solution quickly and reliably.

Part VI

Conclusion & Outlook

To love the journey is to accept no such end. I have found [...] that the most important step a person can take is always the next one.

—*Brandon Sanderson [279]*

OUU is a topic of great significance for domains where robust and dependable solutions are sought or required. This aspect holds particular relevance for industrial sectors such as the defense, automobile, medical, or construction industry, but also for the energy sector, as we have demonstrated in this thesis. With advances in modeling, simulation, and computational capabilities, incorporating uncertainty into the optimization process enhances the realism of the obtained results. Driven by these new demands and possibilities, we introduced three major contributions in this work.

In our first contribution, we presented novel MLMC estimators for variance, standard deviation, and scalarization (a linear combination of mean and standard deviation). The motivation to develop these estimators, stems from their particular importance in OUU, where they enable us to optimize not only statistics such as the mean but also the standard deviation, seeking robust and reliable solutions. Given that the estimators are constructed and evaluated repeatedly throughout the optimization process, it becomes imperative to have precise estimators for the relevant statistics. The standard MLMC estimator, optimized to achieve target precision for the mean, is, however, generally insufficient for accurately estimating these statistics. Therefore, the multilevel resource allocation problem must be adapted to specifically target these alternative statistical objectives. Deriving the variances for these estimators comprises a significant aspect of our work. The new estimators also included algorithmic developments, where we iteratively compute the resource allocation using an analytic approximation and numerical optimization. We furthermore developed three estimators for the covariance term in the scalarization case where we advocated for the use of the Correlation Lift approximation due to its favorable trade-off between approximation quality and computational cost. We showcased sampling results for a four-level test problem called Problem 18, where we conducted a performance comparison between our newly developed estimators and the MLMC estimator for the mean. To assess their efficacy, we utilized a single-level MC estimator as a reference and designed the multilevel resource allocation to achieve a comparable level of accuracy. Our findings demonstrated that our estimators align more closely with the specific SoI, whereas the mean estimator lacks control beyond the variance of the mean estimation. At the same time, we were able to reduce the computational cost compared to classical MC by about a factor of five for this test benchmark.

In our second major contribution, we presented the stochastic optimization method SNOWPAC, which builds upon the derivative-free trust-region method NOWPAC. The method relies on sampling estimators to evaluate the underlying OUU problem. However, this introduces noisy evaluations, where SNOWPAC offers three solutions. First, we employed a noise-adapted trust-region management to restrict the trust-region radius and surrogate model on the noise. Second, we presented GPs as second surrogates to smoothen noise evaluations and effectively decrease the noise. Here, we derived the best root-mean-square estimator to balance the sampling and GP evaluations and showed ways to numerically approximate it. Lastly, since noise evaluations can lead to infeasible solution, we introduced a feasibility-restoration technique to find feasible solutions. Based on these extensions, SNOWPAC was shown to be effective in identifying local optimal solutions even when confronted with noisy black-box evaluations. To validate its performance, we conducted comprehensive benchmark tests. We showed that SNOWPAC converges faster and more accurately to a local optimum than existing methods. Moreover, the newly developed analytic smoothing specifically showed improved results when high accuracy is required (and higher sample numbers are available). Furthermore, we presented two extensions to the method to improve its performance for specific cases: adaptive kernel selection and the use of approximate GPs to handle a large number of evaluation. While SNOWPAC is particularly well-suited for robust optimization problems, its versatility extends beyond this scope. Here, we extended SNOWPAC to another area of application, addressing mixed-integer optimization problems, with a particular focus on neural network architecture design.

Lastly, in our third contribution, we integrated the newly developed MLMC estimators with SNOWPAC in the Dakota software toolkit. We conducted experiments on two benchmark problems to assess the impact of these estimators, where we extended both problems to multilevel cases. The first problem involved a one-dimensional constrained scenario, Problem 18, with one uncertain variable and four levels, while the second problem entailed the two-dimensional constrained Rosenbrock function with three uncertain variables and three levels. Through our analysis, we observed a close alignment between the results obtained using our new estimators with a standard MC reference solution for both benchmarks. At the same time we were able to reduce the computational cost by a factor five for both problems; thus, the cost savings in the sampling directly translates to OUU. In contrast, the standard MLMC estimator for the mean fell short of achieving the desired precision, resulting in suboptimal solutions. In the Rosenbrock example, we also showed that using the MLMC estimator result in a faster convergence and higher accuracy compared to MC when choosing the same computational cost. Finally, we presented result for a wake steering problem with three levels in a realistic and complex application, where we optimized for total power production with reduced variability. We found improved solutions using our new MLMC approach compared to the MC solution—for the same computational cost. We increased total power production for the mean formulation by 20%, whereas we decreased the variation in power production for the scalarized formulation by 5%, compared to the initial design. Hence, both robust formulations achieved their goal. In contrast, the classical MLMC targeting the mean was not able to reduce the objective and performed even worse than the standard MC approach.

Looking ahead, we envision several future directions for our research. Firstly, with regards to MLMC estimators, our objective is to broaden their scope to encompass other formulations that are relevant to robust and reliable optimization problems. This includes exploring CVaR or quantile estimation, both of which can be expressed using sampling estimators. We are encouraged by the initial progress made by Ganesh et al. [127] in this area, which demonstrates promising possibilities. Secondly, our future work regarding SNOWPAC will include investigating the convergence properties of our proposed stochastic derivative-free trust-region framework in terms of reaching first-order critical points. Here, interesting new developments in adaptive sampling [33] and GP posterior consistency results [198] can be useful. Furthermore, considering the growing interest in derivative-free OUU, novel algorithms are emerging (e.g., Dzahini et al. [104, 105], Rinaldi et al. [273]), which provide an opportunity to validate and compare SNOWPAC's performance against these alternative approaches. Lastly, we are also looking forward to apply MLMC estimators for more applications in OUU. Here, the extension to multi-fidelity estimators for higher-order moments is of strong interest. This would enable access to new fields of computationally challenging problems. Additionally, this opens the door to also include new modelling approaches, like data-driven models, as low-cost surrogate models.

Taking all these possible future developments into account, let us end this thesis in the words of Pippin and Gandalf (or J.R.R. Tolkien):

Pippin: "I didn't think it would end this way."
Gandalf: "End? No, the journey doesn't end here.[...]"

J.R.R. Tolkien [305]

Bibliography

[1] Abramowitz, M. and Stegun, I. A., editors (1965). *Handbook of mathematical functions.* General Publishing Company, Ltd., Toronto, Canada.

[2] Abramson, M. A., Audet, C., Chrissis, J. W., and Walston, J. G. (2009). Mesh adaptive direct search algorithms for mixed variable optimization. *Optimization Letters*, 3(1):35–47.

[3] Acerbi, C. and Tasche, D. (2002). Expected shortfall: a natural coherent alternative to Value at Risk. *Economic Notes*, 31(2):379–388.

[4] Adams, B. M., Bohnhoff, W. J., Dalbey, K. R., Ebeida, M. S., Eddy, J. P., Eldred, M. S., Hooper, R. W., Hough, P. D., Hu, K. T., Jakeman, J. D., Khalil, M., Maupin, K. A., Monschke, J. A., Ridgway, E. M., Rushdi, A. A., Seidl, D. T., Stephens, J. A., Swiler, L. P., Tran, A., and Winokur, J. G. (2022). Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis - version 6.16 users manual. Technical report. Updated May, 2022.

[5] Adams, D. (2002). *The Hitchhiker's Guide to the Galaxy.* Gollancz.

[6] Agarwal, A. and Biegler, L. T. (2013). A trust-region framework for constrained optimization using reduced order modeling. *Optimization and Engineering*, 14(1):3–35.

[7] Ahmed, S. and Shapiro, A. (2008). Solving chance-constrained stochastic programs via sampling and integer programming. In *Tutorials in Operations Research.* INFORMS.

[8] Aizerman, A., Braverman, E. M., and Rozoner, L. I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837.

[9] Alexander, S., Coleman, T., and Li, Y. (2006). Minimizing CVaR and VaR for a Portfolio of Derivatives. *Journal of Banking & Finance*, 30(2):583–605.

[10] Ali, M. M., Khompatraporn, C., and Zabinsky, Z. B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of global optimization*, 31(4):635–672.

[11] Alsaleh, A. and Sattler, M. (2019). Comprehensive life cycle assessment of large wind turbines in the US. *Clean Technologies and Environmental Policy*, 21(4):887–903.

[12] A/S, B. A. A. (2012). Denmark-Helicopter Services. `https://www.belair.dk/about-bel-air/`. [Online; accessed 08-August-2023].

[13] Audet, C. (2014). *A survey on direct search methods for blackbox optimization and their applications*. Springer.

[14] Audet, C., Conn, A. R., Le Digabel, S., and Peyrega, M. (2018). A progressive barrier derivative-free trust-region algorithm for constrained optimization. *Computational Optimization and Applications*, 71:307–329.

[15] Audet, C., Custodio, A. L., and Dennis Jr., J. E. (2008). Erratum: mesh addaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 18(4):1501–1503.

[16] Audet, C. and Dennis Jr., J. E. (2002). Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903.

[17] Audet, C. and Dennis Jr., J. E. (2006). Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1):188–217.

[18] Audet, C. and Dennis Jr., J. E. (2009). A progressive barrier for derivative-free nonlinear programming. *SIAM Journal on Optimization*, 20(1):445–472.

[19] Audet, C., Digabel, S. L., Salomon, L., and Tribes, C. (2022a). Constrained blackbox optimization with the nomad solver on the coco constrained test suite. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, page 1683–1690, New York, NY, USA. Association for Computing Machinery.

[20] Audet, C., Dzahini, K. J., Kokkolaras, M., and Le Digabel, S. (2021). Stochastic mesh adaptive direct search for blackbox optimization using probabilistic estimates. *Computational Optimization and Applications*, 79(1):1–34.

[21] Audet, C., Le Digabel, S., Montplaisir, V. R., and Tribes, C. (2022b). Algorithm 1027: NOMAD version 4: Nonlinear Optimization with the MADS algorithm. *ACM Trans. Math. Softw.*, 48(3).

[22] Audet, C., Savard, G., and Zghal, W. (2010). A mesh adaptive direct search algorithm for multiobjective optimization. *European Journal of Operational Research*, 204(3):545–556.

[23] Augustin, F. and Marzouk, Y. M. (2015). NOWPAC: A provably convergent derivative-free nonlinear optimizer with path-augmented constraints.

[24] Augustin, F. and Rentrop, P. (2012). Stochastic Galerkin techniques for random ordinary differential equations. *Numerische Mathematik*.

[25] Ayoul-Guilmard, Q., Ganesh, S., Krumscheid, S., and Nobile, F. (2023). Quantifiying uncertain system outputs via the multi-level monte carlo method—distribution and robustness measures. *International Journal for Uncertainty Quantification*, 13(5):61–98.

[26] Babuška, I. M., Tempone, R., and Zouraris, G. E. (2004). Galerkin finite element approximation of stochastic elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 42(2):800–825.

[27] Babuška, I., Nobile, F., and Tempone, R. (2010). A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Review*, 52(2):317–355.

[28] Bachoc, F. (2013). Cross validation and maximum likelihood estimations of hyperparameters of gaussian processes with model misspecification. *Computational Statistics & Data Analysis*, 66:55–69.

[29] Balas, E., Ceria, S., and Cornuéjols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical programming*, 58(1-3):295–324.

[30] Bandeira, A. S., Scheinberg, K., and Vicente, L. N. (2014). Convergence of trust-region methods based on probabilistic models. *SIAM Journal on Optimization*, 24(3):1238–1264.

[31] Barth, A., Schwab, C., and Zollinger, N. (2011). Multi-level Monte Carlo Finite Element method for elliptic PDEs with stochastic coefficients. *Numerische Mathematik*, 119(1):123–161.

[32] Bayraksan, G. and Morton, D. P. (2006). Assessing solution quality in stochastic programs. *Mathematical Programming, Series B*, 108:495–514.

[33] Beiser, F., Keith, B., Urbainczyk, S., and Wohlmuth, B. (2020). Adaptive sampling strategies for risk-averse stochastic optimization with constraints. *arXiv preprint arXiv:2012.03844*.

[34] Bergstra, J., Yamins, D., and Cox, D. D. (2013a). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures.

[35] Bergstra, J., Yamins, D., and Cox, D. D. (2013b). Tree Structured Parzen's Estimator. `https://github.com/hyperopt/hyperopt`. [Online; accessed 10-May-2023].

[36] Bertsekas, D. P. (1999). *Nonlinear Programming.* Athena Scientific.

[37] Bespalov, A., Praetorius, D., Rocchi, L., and Ruggeri, M. (2019). Convergence of adaptive stochastic galerkin fem. *SIAM Journal on Numerical Analysis*, 57(5):2359–2382.

[38] Bhatnagar, S., Prasad, H. L., and Prashanth, L. A. (2013). *Stochastic recursive algorithms for optimization*, volume 434 of *Lecture notes in control and information sciences*. Springer-Verlag London Heidelberg New York Dordrecht.

[39] Bienstock, D. (1996). Computational study of a family of mixed-integer quadratic programming problems. *Mathematical programming*, 74(2):121–140.

[40] Bierig, C. and Chernov, A. (2016). Estimation of arbitrary order central statistical moments by the multilevel monte carlo method. *Stochastics and Partial Differential Equations Analysis and Computations*, 4(1):3–40.

[41] Bijl, H., Schön, T. B., van Wingerden, J.-W., and Verhaegen, M. (2017). System identification through online sparse gaussian process regression with input noise. *IFAC Journal of Systems and Control*, 2:1–11.

[42] Bijl, H., van Wingerden, J.-W., Schön, T. B., and Verhaegen, M. (2015). Online Sparse Gaussian Process Regression Using FITC and PITC Approximations. *IFAC-PapersOnLine*, 48(28):703–708.

[43] Billionnet, A. and Elloumi, S. (2007). Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Mathematical Programming*, 109(1):55–68.

[44] Billionnet, A., Elloumi, S., and Lambert, A. (2012). Extending the qcr method to general mixed-integer programs. *Mathematical programming*, 131(1-2):381–401.

[45] Biondini, G. (2015). Chapter 2 - an introduction to rare event simulation and importance sampling. In Govindaraju, V., Raghavan, V. V., and Rao, C., editors, *Big Data Analytics*, volume 33 of *Handbook of Statistics*, pages 29–68. Elsevier.

[46] Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

[47] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[48] Bohachevsky, I. O., Johnson, M. E., and Stein, M. L. (1986). Generalized simulated annealing for function optimization. *Technometrics*, 28(3):209–217.

[49] Bonami, P. and Lee, J. (2007). Bonmin user manual. *Numer Math*, 4:1–32.

[50] Boob, D., Deng, Q., and Lan, G. (2023). Stochastic first-order methods for convex and nonconvex functional constrained optimization. *Mathematical Programming*, 197(1):215–279.

[51] Borchers, B. and Mitchell, J. E. (1994). An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers & Operations Research*, 21(4):359–367.

[52] Bortz, D. M. and Kelley, C. T. (1998). *Computational methods for optimal design and control*, volume 24 of *Progress in Systems and Control Theory*, chapter The simplex gradient and noisy optimization problems, pages 77–90. de Gruyter.

[53] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA. ACM.

[54] Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311.

[55] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press, New York, NY, USA.

[56] Bungartz, H.-J. and Griebel, M. (2004). Sparse grids. *Acta Numerica*, 13:147–269.

[57] Caflisch, R. E. (1998). Monte carlo and quasi-monte carlo methods. *Acta Numerica*, 7:1–49.

[58] Cameron, R. H. and Martin, W. T. (1947). The orthogonal development of non-linear functionals in series of Fourier-Hermite functionals. *The Annals of Mathematics, 2nd Ser.*, 48(2):385–392.

[59] Cao, Y., Brubaker, M. A., Fleet, D. J., and Hertzmann, A. (2015). Efficient Optimization for Sparse Gaussian Process Regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(12):2415–2427.

[60] Carere, G., Strazzullo, M., Ballarin, F., Rozza, G., and Stevenson, R. (2021). A weighted pod-reduction approach for parametrized pde-constrained optimal control problems with random inputs and applications to environmental sciences. *Computers & Mathematics with Applications*, 102:261–276.

[61] Carter, R. G. (1991). On the global convergence of trust region algorithms using inexact gradient information. *SIAM Journal on Numerical Analysis*, 28(1):251–265.

[62] Cawley, G. C. and Talbot, N. L. C. (2007). Preventing over-fitting during model selection via Bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861.

[63] Chalupka, K., Williams, C. K. I., and Murray, I. (2013). A Framework for Evaluating Approximation Methods for Gaussian Process Regression. *Journal of Machine Learning Research*, 14:333–350.

[64] Chang, K. H., Hong, L. J., and Wan, H. (2013). Stochasic trust-region response-surface method (STRONG) - a new response-surface framework for simulation optimization. *INF*, 25(2):230–243.

[65] Chen, P., Quarteroni, A., and Rozza, G. (2017). Reduced basis methods for uncertainty quantification. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):813–869.

[66] Chen, P. and Schwab, C. (2017). *Model Order Reduction Methods in Computational Uncertainty Quantification*, pages 937–990. Springer International Publishing, Cham.

[67] Chen, R., Menickelly, M., and Scheinberg, K. (2018). Stochastic optimization using a trust-region method and random models. *Mathematical Programming*, 169(2):447–487.

[68] Choi, T. D. and Kelley, C. T. (2000). Superlinear convergence and implicit filtering. *SIAM Journal on Optimization*, 10(4):1149–1162.

[69] Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2015). The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204.

[70] Churchill, W. (1898). *The story of the Malakand Field Force: an episode of frontier war*. Longmans, Green.

[71] Clausen, J. (1999). Branch and bound algorithms-principles and examples. *Department of Computer Science, University of Copenhagen*, pages 1–30.

[72] Cliffe, K. A., Giles, M. B., Scheichl, R., and Teckentrup, A. L. (2011). Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients. *Computing and Visualization in Science*, 14(1):3.

[73] Collier, N., Haji-Ali, A.-L., Nobile, F., von Schwerin, E., and Tempone, R. (2015). A continuation multilevel Monte Carlo algorithm. *BIT Numerical Mathematics*, 55(2):399–432.

[74] Conejo, P., Karas, E. W., and Pedroso, L. G. (2015). A trust-region derivative-free algorithm for constrained optimization. *Optimization Methods and Software*, 30(6):1126–1145.

[75] Conn, A. R. and Digabel, S. L. (2013). Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optimization Methods and Software*, 28(1):139–158.

[76] Conn, A. R., Gould, N. I. M., Sartenaer, A., and Toint, P. L. (1993a). Global convergence of a class of trust region algorithms for optimization using inexact projections on convex constraints. *SIAM Journal on Optimization*, 3(1):164–221.

[77] Conn, A. R., Gould, N. I. M., Sartenaer, A., and Toint, P. L. (1993b). Global convergence of a class of trust region algorithms for optimization using inexact projections on convex constraints. *SIAM Journal on Optimization*, 3(1):164–221.

[78] Conn, A. R., Gould, N. I. M., and Toint, P. L. (2000). *Trust Region Methods.* MPS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics.

[79] Conn, A. R., Scheinberg, K., and Toint, P. L. (1997). On the convergence of derivative-free methods for unconstrained optimization. *Approximation theory and optimization: tributes to MJD Powell*, pages 83–108.

[80] Conn, A. R., Scheinberg, K., and Vicente, L. N. (2009a). Global convergence of general derivative-free trust-region algorithms to first- and second-order critical points. *SIAM Journal on Optimization*, 20(1):387–415.

[81] Conn, A. R., Scheinberg, K., and Vicente, L. N. (2009b). *Introduction to Derivative-Free Optimization.* MPS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics and the Mathematical Programming Society.

[82] Conn, A. R., Scheinberg, K., and Vicente, L. N. (2009c). *Introduction to derivative-free optimization.* SIAM, Society for Industrial and Applied Mathematics and the Mathematical Programming Society, Philadelphia.

[83] Contreras Montoya, L. T., Hayyani, M. Y., Issa, M., Ilinca, A., Ibrahim, H., and Rezkallah, M. (2021). 8 - wind power plant planning and modeling. In Kabalci, E., editor, *Hybrid Renewable Energy Systems and Microgrids*, pages 259–312. Academic Press.

[84] Cornford, D., Nabney, I., and Williams, C. (2002). Modelling frontal discontinuities in wind fields. *Journal of Nonparametric Statistics*, 14.

[85] Cornuejols, G. and Tütüncü, R. (2007). *Optimization Methods in Finance.* Cambridge University Press.

[86] Coutsias, E. A., Seok, C., and Dill, K. A. (2004). Using quaternions to calculate rmsd. *Journal of Computational Chemistry*, 25(15):1849–1857.

[87] C.Raffel (2017). Bayesian Optimization. `https://github.com/craffel/simple_spearmint`. [Online; accessed 10-May-2023].

[88] Craven, B. D. and Islam, S. M. N. (2005). *Optimization in Economics and Finance*, volume 7 of *Dynamic Modeling and Econometrics in Economics and Finance.* Springer-Verlag, Berlin/Heidelberg.

[89] Currie, J. OPTI Toolbox. `https://github.com/jonathancurrie/OPTI`. [Online; accessed 10-May-2023].

[90] Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. *The computer journal*, 8(3):250–255.

[91] Dalbey, K. R., Eldred, M. S., Geraci, G., Jakeman, J. D., Maupin, K. A., Monschke, J. A., Seidl, D. T., Swiler, L. P., Tran, A., with Menhorn, F., and Zeng, X. (2022).

Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis - version 6.16 theory manual. Technical report. Updated May, 2022.

[92] Dantzig, G. B. and Thapa, M. N. (1997). *Linear Programming 1: Introduction.* Springer-Verlag New York, Inc.

[93] Dantzig, G. B. and Thapa, M. N. (2003). *Linear Programming 2: Theory and Extension.* Springer-Verlag New York, Inc.

[94] David, H. A. and Nagaraja, H. N. (2003). *Order statistics.* John Wiley & Sons, Inc., Hoboken, New Jersey, 3rd edition.

[95] de Klerk, E. (2010). *Stories about Maxima and Minima.* Tilburg University Press, Tilburg.

[96] Detommaso, G., Dodwell, T., and Scheichl, R. (2019). Continuous level monte carlo and sample-adaptive model hierarchies. *SIAM/ASA Journal on Uncertainty Quantification*, 7(1):93–116.

[97] Dewancker, I., McCourt, M., Clark, S., Hayes, P., Johnson, A., and Ke, G. (2016). A stratified analysis of bayesian optimization methods.

[98] Di Pillo, G., Lucidi, S., and Rinaldi, F. (2013). A derivative-free algorithm for constrained global optimization based on exact penalty functions. *Journal of Optimization Theory and Applications*, Springer Science+Business Media New York.

[99] Dick, J., Kuo, F. Y., and Sloan, I. H. (2013). High-dimensional integration: The quasi-monte carlo way. *Acta Numerica*, page 133–288.

[100] Dierkes, U., Hildebrandt, S., and Sauvigny, F. (2010). *Minimal Surfaces.* Springer – A Series of Comprehensive Studies in Mathematics.

[101] Dodge, Y. and Rousson, V. (1999). The complications of the fourth central moment. *The American Statistician*, 53(3):267–269.

[102] Durrande, N., Ginsbourger, D., and Roustant, O. (2012). Additive covariance kernels for high-dimensional gaussian process modeling. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 21, pages 481–499.

[103] Duvenaud, D. (2014). *Automatic model construction with Gaussian processes.* PhD thesis, University of Cambridge.

[104] Dzahini, K. J., Kokkolaras, M., and Le Digabel, S. (2023). Constrained stochastic blackbox optimization using a progressive barrier and probabilistic estimates. *Mathematical Programming*, 198(1):675–732.

[105] Dzahini, K. J. and Wild, S. M. (2022). Stochastic trust-region algorithm in random subspaces with convergence and expected complexity analyses.

[106] Efron, B. and Hastie, T. (2016). *Computer Age Statistical Inference.* Cambridge University Press.

[107] Efron, B. and Tibshirani, R. (1994). *An introduction to the bootstrap.* Chapman & Hall.

[108] Eigel, M., Gittelson, C. J., Schwab, C., and Zander, E. (2014). Adaptive stochastic galerkin fem. *Computer Methods in Applied Mechanics and Engineering*, 270:247–269.

[109] Einstein, A., Jeffery, G., and Perrett, W. (1923). *Sidelights on Relativity.* E.P. Dutton.

[110] Eldread, M. S., Geraci, G., Gorodetsky, A., and Jakeman, J. D. (2018). Multilevel-multidelity approaches for forward uq in the darpa sequoia project. In *2018 AIAA Non-Deterministic Approaches Conference, AIAA SciTech Forum.*

[111] Elfverson, D., Hellman, F., and Målqvist, A. (2016). A multilevel monte carlo method for computing failure probabilities. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):312–330.

[112] Eriksson, D., Bindel, D., and Shoemaker, C. (2015). Surrogate Optimization Toolbox (pySOT). `https://github.com/dme65/pySOT`. [Online; accessed 10-May-2023].

[113] Eriksson, D. and Poloczek, M. (2021). Scalable constrained bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 730–738. PMLR.

[114] Ernst, O. G. and Ullmann, E. (2010). Stochastic galerkin matrices. *SIAM Journal on Matrix Analysis and Applications*, 31(4):1848–1872.

[115] Euler, L. (1952). *Methodus inveniendi lineas curvas : maximi minimive proprietate gaudentes sive solutio problematis isoperimetrici latissimo sensu accepti.* Orell Füssli Turici.

[116] Farcas, I.-G. (2020). *Context-aware Model Hierarchies for Higher-dimensional Uncertainty Quantification.* PhD thesis, Technische Universität München.

[117] Farcas, I.-G., Latz, J., Ullmann, E., Neckel, T., and Bungartz, H.-J. (2020). Multilevel adaptive sparse leja approximations for bayesian inverse problems. *SIAM Journal on Scientific Computing*, 42(1):A424–A451.

[118] Farcaş, I.-G., Sârbu, P. C., Bungartz, H.-J., Neckel, T., and Uekermann, B. (2018). Multilevel adaptive stochastic collocation with dimensionality reduction. In *Sparse Grids and Applications-Miami 2016*, pages 43–68. Springer.

[119] Farcaș, I.-G., Peherstorfer, B., Neckel, T., Jenko, F., and Bungartz, H.-J. (2023). Context-aware learning of hierarchies of low-fidelity models for multi-fidelity uncertainty quantification. *Computer Methods in Applied Mechanics and Engineering*, 406:115908.

[120] Fermi, E. and Metropolis, N. (1952). Los alamos unclassified report ls–1492. *Rapport technique, Los Alamos National Laboratory, Los Alamos, New Mexico.*

[121] Fernández-Godino, M. G. (2023). Review of multi-fidelity models.

[122] Fletcher, R., Gould, N. I. M., Leyffer, S., Toint, P. L., and Wächter, A. (2002). Global convergence of a trust-region sqp-filter algorithm for general nonlinear programming. *SIAM Journal on Optimization*, 13(3):635–659.

[123] Fletcher, R. and Leyffer, S. (2002). Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239–269.

[124] Frazier, P., Powell, W., and Dayanik, S. (2009). The knowledge-gradient policy for correlated nonorm beliefs. *INFORMS Journal on Computing*, 21(4):599–613.

[125] Frederick, J., Conley, E., Nole, M., Marchitto, T.-e., and Wagman, B. (2022). Quantifying the known unknown: Including marine sources of greenhouse gases in climate modeling. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

[126] Gal, Y. and Turner, R. (2015). Improving the gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 655–664. JMLR.org.

[127] Ganesh, S. and Nobile, F. (2022). Gradient-based optimisation of the conditional-value-at-risk using the multi-level monte carlo method.

[128] García-Palomares, U. M., García-Urrea, I. J., and Rodríguez-Hernández, P. S. (2013). On sequential and parallel non-monotone derivative-free algorithms for box constrained optimization. *Optimization Methods and Software*, 28(6):1233–1261.

[129] Gardner, J. R., Kusner, M. J., Xu, Z., Weinberger, K. Q., and Cunningham, J. P. (2014). Bayesian optimization with inequality constraints. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–937–II–945. JMLR.org.

[130] Gauss, C. F. (1863). *Carl Friedrich Gauss Werke.* Königliche Gesellschaft der Wissenschaften, Göttingen.

[131] Gavana, A. (2013). Infinity 77. `http://infinity77.net/global_optimization/test_functions_1d.html` [Online; accessed 04-July-2023].

[132] Geiersbach, C. and Wollner, W. (2020). A stochastic gradient method with mesh refinement for pde-constrained optimization under uncertainty. *SIAM Journal on Scientific Computing*, 42(5):A2750–A2772.

[133] Geraci, G., Eldred, M. S., and Iaccarino, G. (2017). A multifidelity multilevel monte carlo method for uncertainty propagation in aerospace applications. In *19th AIAA Non-Deterministic Approaches Conference*, page 1951.

[134] Geraci, G., Menhorn, F., Huan, X., Safta, C., Marzouk, Y., Najm, H., and Eldred, M. (2019). Progress in scramjet design optimization under uncertainty using simulations of the hifire configuration.

[135] Ghanem, R. G. and Spanos, P. D. (1991). *Stochastic finite elements: A spectral approach*. Springer-Verlag New York.

[136] Giles, M. B. (2008). Multilevel Monte Carlo Path Simulation. *Operations Research*, 56(3):607–617.

[137] Giles, M. B. (2015). Multilevel monte carlo methods. *Acta Numerica*, 24:259–328.

[138] Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1998). NPSOL 6.2—A Fortran package for nonlinear programming. Technical report, Systems Optimization Laboratoy Technical Report SOL 86-1.

[139] Gilmore, P. and Kelley, C. T. (1995). An implicit filtering algorithm for optimization of functions with many local minima. *SIAM Journal on Optimization*, 5(2):269–285.

[140] Goda, T., Hironaka, T., and Iwamoto, T. (2020). Multilevel monte carlo estimation of expected information gains. *Stochastic Analysis and Applications*, 38(4):581–600.

[141] Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA.

[142] Gorodetsky, A. A., Geraci, G., Eldred, M. S., and Jakeman, J. D. (2020a). A generalized approximate control variate framework for multifidelity uncertainty quantification. *Journal of Computational Physics*, 408:109257.

[143] Gorodetsky, A. A., Jakeman, J. D., Geraci, G., and Eldred, M. S. (2020b). Mfnets: multi-fidelity data-driven networks for bayesian learning and prediction. *International Journal for Uncertainty Quantification*, 10(6).

[144] Gould, N. I. M. and Orban, D. CUTEst– A Constrained and Unconstrained Testing Environment on Steroids. `https://ccpforge.cse.rl.ac.uk/gf/project/cutest/wiki/`, [Online; accessed 05-June-2023].

[145] Gould, N. I. M. and Toint, P. L. (2010). Nonlinear programming without a penalty function or a filter. *Mathematical Programming*, 122(1):155–196.

[146] Gramacy, R. B., Gray, G. A., Digabel, S. L., Lee, H. K. H., Ranjan, P., Wells, G., and Wild, S. M. (2016). Modeling an augmented lagrangian for blackbox constrained optimization. *Technometrics*, 58(1):1–11.

[147] Gratton, S., Royer, C. W., Vicente, L. N., and Zhang, Z. (2018). Complexity and global rates of trust-region methods based on probabilistic models. *IMA Journal of Numerical Analysis*, 38(3):1579–1597.

[148] Grippo, L. and Sciandrone, M. (2007). Nonmonotone derivate–free methods for nonlinear equations.

[149] Groetsch, C. (1999). *Inverse Problems: Activities for Undergraduates*. The Mathematical Associtation of America.

[150] Gupta, O. K. and Ravindran, A. (1985). Branch and bound experiments in convex nonlinear integer programming. *Management science*, 31(12):1533–1546.

[151] Haji-Ali, A.-L., Nobile, F., Tamellini, L., and Tempone, R. (2016). Multi-index stochastic collocation for random pdes. *Computer Methods in Applied Mechanics and Engineering*, 306:95–122.

[152] Hales, T. C. (2005). A proof of the Kepler conjecture. *Annals of Mathematics*, 162:1065–1185.

[153] Hassanien, A.-E., Grosan, C., and Fahmy Tolba, M. (2016). *Applications of Intelligent Optimization in Biology and Medicine*, volume 96 of *Intelligent Systems Reference Library*. Springer International Publishing, Cham.

[154] Heinkenschloss, M. and Vicente, L. N. (2002). Analysis of inexact trust-region SQP algorithms. *SIAM Journal on Optimization*, 12(2):283–302.

[155] Heinrich, S. (2001). Multilevel monte carlo methods. In Margenov, S., Waśniewski, J., and Yalamov, P., editors, *Large-Scale Scientific Computing*, pages 58–67, Berlin, Heidelberg. Springer Berlin Heidelberg.

[156] Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, page 282–290, Arlington, Virginia, USA. AUAI Press.

[157] Hock, W. and Schittkowski, K. (1980). Test Examples for Nonlinear Programming Codes. *Journal of Optimization Theory and Applications*, 30(1):127–129.

[158] Hock, W. and Schittkowski, K. (1981). *Lecture Notes in Economics and Mathematical Systems*, chapter Test examples for nonlinear programming, no. 187. Springer.

[159] Hofmann, T., Schölkopf, B., and Smola, A. J. (2008). Kernel methods in machine learning. *Ann. Statist.*, 36(3):1171–1220.

[160] Hooke, R. and Jeeves, T. A. (1961). Direct search solution of numerical and statistical problems. *Journal of the ACM*, 8(2):212–229.

[161] Howard, A. A., Perego, M., Karniadakis, G. E., and Stinis, P. (2022). Multifidelity deep operator networks.

[162] Hsieh, A., Maniaci, D. C., Herges, T. G., Geraci, G., Seidl, D. T., Eldred, M. S., Blaylock, M. L., and Houchens, B. C. (2020). Multilevel uncertainty quantification using cfd and openfast simulations of the swift facility. In *AIAA Scitech 2020 Forum*, page 1949.

[163] Hu, J. and Jin, S. (2016). A stochastic galerkin method for the boltzmann equation with uncertainty. *Journal of Computational Physics*, 315:150–168.

[164] Hutter, F., Hoos, H. H., and Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. `https://github.com/automl/SMAC3`. [Online; accessed 10-May-2023].

[165] Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer.

[166] Ilievski, I., Akhtar, T., Feng, J., and Shoemaker, C. A. (2017). Efficient hyperparameter optimization for deep learning algorithms using deterministic rbf surrogates. In *AAAI*, pages 822–829.

[167] Ivor, T. (1941). *Greek Mathematical Works.* Harvard University Press, Cambridge, first edition.

[168] Jakeman, J. D., Eldred, M. S., Geraci, G., and Gorodetsky, A. (2020). Adaptive multi-index collocation for uncertainty quantification and sensitivity analysis. *International Journal for Numerical Methods in Engineering*, 121(6):1314–1343.

[169] Jasra, A., Law, K., and Suciu, C. (2020). Advanced multilevel monte carlo methods. *International Statistical Review*, 88(3):548–579.

[170] Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492.

[171] Jonkman, J., Butterfield, S., Musial, W., and Scott, G. (2009). Definition of a 5-mw reference wind turbine for offshore system development. Technical report, National Renewable Energy Laboratory NREL/TP-500-38060.

[172] Jr., J. D. and Torczon, V. (1994). *Derivative-free pattern search methods for multidisciplinary design problems.* AIAA Meeting Paper.

[173] Kannan, A. and Wild, S. M. (2012). Obtaining Quadratic Models of Noisy Functions. Technical report, Argonne National Laboratory, Illinois.

[174] Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T. (2010). Gaussian processes for object categorization. *International journal of computer vision*, 88:169–188.

[175] Kelley, C. T. (1999). *Iterative methods for optimization.* SIAM, Society for Industrial and Applied Mathematics, Philadelphia.

[176] Kelley, C. T. (2011). *Implicit filtering.* SIAM.

[177] Kibzun, A. and Kan, Y. (1996). *Stochastic programming problems: with probability and quantile functions.* John Wiley & Sons Ltd.

[178] Kibzun, A. and Uryasev, S. (1998). Differentiability of probability function. *Stochastic Analysis and Applications*, 16(6):1101–1128.

[179] Kiefer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Ma*, 23(3):462–466.

[180] Kim, S., Pasupathy, R., and Henderson, S. G. (2011). A guide to sample-average approximation. http://people.orie.cornell.edu/shane/pubs/SAAGuide.pdf.

[181] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[182] Kislaya, R., Menhorn, F., Meinhard, T., Leal-Taixe, L., and Bungartz, H.-J. (2019). Neural network hyperparameter optimization using snowpac. Master's thesis, Technical University of Munich.

[183] Kitti, M. History of Optimization. `http://www.mitrikitti.fi/opthist.html`, [Online; accessed 05-June-2023].

[184] Kleijnen, J. P. C., Ridder, A. A. N., and Rubinstein, R. Y. (2013). *Variance Reduction Techniques in Monte Carlo Methods*, pages 1598–1610. Springer US, Boston, MA.

[185] König, H. (1986). *Eigenvalue Distribution of Compact Operators*, volume 16 of *Operator Theory: Advances and Applications.* Birkhäuser Basel, Basel.

[186] Krejić, N. and Jerinkić, N. K. (2014). Stochastic gradient methods for unconstrained optimization. *Pesquisa Operacional*, 34:373–393.

[187] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.

[188] Krumscheid, S. and Nobile, F. (2018). Multilevel monte carlo approximation of functions. *SIAM/ASA Journal on Uncertainty Quantification*, 6(3):1256–1293.

[189] Krumscheid, S., Nobile, F., and Pisaroni, M. (2020). Quantifying uncertain system outputs via the multilevel monte carlo method - part i: Central moment estimation. *Journal of Computational Physics*, 414:109466.

[190] Kushner, H. J. and Yin, G. G. (1997). *Stochastic approximation algorithms and applications*, volume 35 of *Applications of mathematics.* Springer Verlag New York.

[191] Larson, J. and Billups, S. C. (2016). Stochastic derivative-free optimization using a trust region framework. *Computational Optimization and applications*, 64:619–645.

[192] Larson, J., Menickelly, M., and Wild, S. M. (2019). Derivative-free optimization methods. *Acta Numerica*, 28:287–404.

[193] Lázaro-Gredilla, M., Quiñonero-Candela, J., Com, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. (2010). Sparse Spectrum Gaussian Process Regression. *Journal of Machine Learning Research*, 11:1865–1881.

[194] Le Digabel, S. (2011). Algorithm 909: NOMAD: nonlinear optimization with the MADS algorithm. *ACM Trans. Math. Softw.*, 37(4):44:1–44:15.

[195] Le Digabel, S. (2011). NOMAD: Nonlinear optimization with the MADS algorithm. *ACM Trans. Math. Softw.*, 37:44.

[196] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[197] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

[198] Lederer, A., Umlauft, J., and Hirche, S. (2019). Uniform error bounds for gaussian process regression with application to safe control. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[199] Lehel, C. and Opper, M. (2002). Sparse online gaussian processes. *Neural Computation.*

[200] Letham, B., Karrer, B., Ottoni, G., and Bakshy, E. (2019). Constrained bayesian optimization with noisy experiments. *Bayesian Anal.*, 14(2):495–519.

[201] Leweke, T., Quaranta, H. U., Bolnot, H., Blanco-Rodríguez, F. J., and Dizès, S. L. (2014). Long- and short-wave instabilities in helical vortices. *Journal of Physics: Conference Series*, 524(1):012154.

[202] Lewis, R. M. and Torczon, V. (1999). Pattern search algorithms for bound constrained minimization. *SIAM Journal on optimization*, 9(4):1082–1099.

[203] Lewis, R. M. and Torczon, V. (2000). Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization*, 10(3):917–941.

[204] Leyffer, S. and Mahajan, A. (2010). Nonlinear constrained optimization: methods and software. *Argonee National Laboratory, Argonne, Illinois*, 60439.

[205] Li, J., Li, J., and Xiu, D. (2011). An efficient surrogate-based method for computing rare failure probability. *Journal of Computational Physics*, 230:8683–8697.

[206] Li, J. and Xiu, D. (2010). Evaluation of failure probability via surrogate models. *Journal of Computational Physics*, 229:8966–8980.

[207] Li, J. and Xiu, D. (2012). Computation of failure probability subject to epistemic uncertainty. *SIAM Journal on Scientific Computing*, 34(6):A2946–A2964.

[208] Liu, H., Ong, Y.-S., Shen, X., and Cai, J. (2020). When gaussian process meets big data: A review of scalable gps. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11):4405–4423.

[209] Liu, X. (2012). *Appendix A: The Delta Method*, pages 405–406. John Wiley & Sons, Ltd.

[210] Liuzzi, G., Lucidi, S., and Sciandrone, M. (2006). A derivative-free algorithm for linearly constrained finite minimax problems. *SIAM Journal on Optimization*, 16:1054–1075.

[211] Liuzzi, G., Lucidi, S., and Sciandrone, M. (2010). Sequential penalty derivative-free methods for nonlinear constrained optimization. *SIAM Journal on Optimization*, 20(5):2614–2635.

[212] Lucidi, S. and Sciandrone, M. (2002). On the global convergence of derivative-free methods for unconstrained optimization. *SIAM Journal on Optimization*, 13(1):97–116.

[213] Ma, X. and Zabaras, N. (2009). An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. *Journal of Computational Physics*, 228(8):3084–3113.

[214] Ma, X. and Zabaras, N. (2010). An adaptive high-dimensional stochastic model representation technique for the solution of stochastic partial differential equations. *Journal of Computational Physics*, 229(10):3884–3915.

[215] MacKay, D. J. et al. (1998). Introduction to gaussian processes. *NATO ASI series F computer and systems sciences*, 168:133–166.

[216] Manzhos, S. and Ihara, M. (2023). Optimization of hyperparameters of Gaussian process regression with the help of a low-order high-dimensional model representation: application to a potential energy surface. *Journal of Mathematical Chemistry*, 61(1):7–20.

[217] March, A. and Willcox, K. (2012). Constrained multifidelity optimization using model calibration. *Structural and Multidisciplinary Optimization*, 46:93–109.

[218] Martino, L., Laparra, V., and Camps-Valls, G. (2017). Probabilistic cross-validation estimators for gaussian process regression. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 823–827.

[219] Matthies, H. G. and Keese, A. (2005). Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194:1295–1331.

[220] Melkumyan, A. and Ramos, F. (2011). Multi-kernel gaussian processes. In *Twenty-second international joint conference on artificial intelligence*.

[221] Menhorn, F., Augustin, F., Bungartz, H. J., and Marzouk, Y. M. (2022). A trust-region method for derivative-free nonlinear constrained stochastic optimization.

[222] Menhorn, F., Augustin, F., and Marzouk, Y. M. (2017). Derivative-free constrained stochastic optimization of a scramjet, using snowpac.

[223] Menhorn, F., Geraci, G., Seidl, D. T., Eldred, M. S., King, R., Bungartz, H.-J., and Marzouk, Y. (2020). Higher moment multilevel estimators for optimization under uncertainty applied to wind plant design. In *AIAA Scitech 2020 Forum*, page 1952.

[224] Menhorn, F., Geraci, G., Seidl, D. T., Marzouk, Y. M., Eldred, M. S., and Bungartz, H.-J. (2024). Multilevel monte carlo estimators for derivative-free optimization under uncertainty. *International Journal for Uncertainty Quantification*, 14(3).

[225] Merriman, M. (1877). On the history of the method of least squares. *The Analyst*, 4(2):33–36.

[226] Metropolis, N. and Ulam, S. (1949). The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341.

[227] Meza, J. C., Hough, P. D., Williams, P. J., and Oliva, R. A. (2022). OPT++ 2.4—A Nonlinear Optimization Package in C++.

[228] Mockus, J. (1974). On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference Novosibirsk*, volume 27 of *Lecture Notes in Computer Science*, pages 400–404. Springer-Verlag Berlin.

[229] Mockus, J. (1989). *Trust Region Methods*. Mathematics and its Applications. Springer Netherlands.

[230] Mood, A., Graybill, F., and Boes, D. (1974). *Introduction to the Theory of Statistics*. International Student edition. McGraw-Hill.

[231] Moré, J. J. (1983). *Recent Developments in Algorithms and Software for Trust Region Methods*, pages 258–287. Springer Berlin Heidelberg, Berlin, Heidelberg.

[232] Moré, J. J. and Wild, S. M. (2009). Benchmarking Derivative-Free Optimization Algorithms. *SIAM Journal on Optimization*, 20(1):172–191.

[233] Motamed, M. (2020). A multi-fidelity neural network surrogate sampling method for uncertainty quantification. *International Journal for Uncertainty Quantification*, 10(4).

[234] Murty, P. (2017). Chapter 24 - renewable energy sources. In Murty, P., editor, *Electrical Power Systems*, pages 783–800. Butterworth-Heinemann, Boston.

[235] Naish-Guzman, A. and Holden, S. (2007). The Generalized FITC Approximation. In *Advances in Neural Information Processing Systems 20*. Twenty-First Annual Conference on Neural Information Processing Systems.

[236] Narayan, A., Gittelson, C., and Xiu, D. (2014). A stochastic collocation algorithm with multifidelity models. *SIAM Journal on Scientific Computing*, 36(2):A495–A521.

[237] Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4):308–313.

[238] Newby, E. (2013). *General solution methods for mixed integer quadratic programming and derivative free mixed integer non-linear programming problems.* PhD thesis.

[239] Newby, E. and Ali, M. M. (2015). A trust-region-based derivative free algorithm for mixed integer programming. *Computational Optimization and Applications*, 60(1):199–229.

[240] Ng, L. W. and Willcox, K. E. (2014). Multifidelity approaches for optimization under uncertainty. *International Journal for numerical methods in Engineering*, 100(10):746–772.

[241] Ng, L. W.-T. and Eldred, M. (2012). Multifidelity uncertainty quantification using non-intrusive polynomial chaos and stochastic collocation. In *53rd aiaa/asme/asce/ahs/asc structures, structural dynamics and materials conference 20th aiaa/asme/ahs adaptive structures conference 14th aiaa*, page 1852.

[242] Nobile, F., Tempone, R., and Webster, C. G. (2008). A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 46(5):2309–2345.

[243] NREL (2022a). FLORIS. Version 3.4.0. `https://github.com/NREL/floris`, [Online; accessed 04-July-2023].

[244] NREL (2022b). Floris Wake Models. `https://nrel.github.io/floris/wake_models.html`, [Online; accessed 04-July-2023].

[245] of Alexandria, E., Fitzpatrick, R., and Heiberg, J. (2007). *Euclid's Elements of Geometry.* Richard Fitzpatrick.

[246] Ökten, G. and Liu, Y. (2021). Randomized quasi-monte carlo methods in global sensitivity analysis. *Reliability Engineering & System Safety*, 210:107520.

[247] Omelyan, I. and Kovalenko, A. (2019). Enhanced solvation force extrapolation for speeding up molecular dynamics simulations of complex biochemical liquids. *The Journal of Chemical Physics*, 151(21):214102.

[248] O'Neill, B. (2014). Some useful moment results in sampling problems. *The American Statistician*, 68(4):282–296.

[249] Parkinson, A., Balling, R., and Hedengren, J. (2013). *Optimization Methods for Engineering Design.* Brigham Young University.

[250] Peherstorfer, B., Cui, T., Marzouk, Y., and Willcox, K. (2016a). Multifidelity importance sampling. *Computer Methods in Applied Mechanics and Engineering*, 300:490–509.

[251] Peherstorfer, B., Willcox, K., and Gunzburger, M. (2016b). Optimal model management for multifidelity monte carlo estimation. *SIAM Journal on Scientific Computing*, 38(5):A3163–A3194.

[252] Peherstorfer, B., Willcox, K., and Gunzburger, M. (2018). Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *Siam Review*, 60(3):550–591.

[253] Pellegrini, R., Serani, A., Liuzzi, G., Rinaldi, F., Lucidi, S., and Diez, M. (2022). A derivative-free line-search algorithm for simulation-driven design optimization using multi-fidelity computations. *Mathematics*, 10(3):481.

[254] Petersen, K. B. and Pedersen, M. S. (2012). The matrix cookbook. `http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf`, Version 11/15/2012, [Online; accessed 05-June-2023].

[255] Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497.

[256] Powell, M. (2001). On the lagrange functions of quadratic models that are defined by interpolation. *Optimization Methods and Software*, 16(1-4):289–309.

[257] Powell, M. (2002). UOBYQA: Unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92(3):555–582.

[258] Powell, M. (2004). Least Frobenius norm updating of quadratic models that satisfy interpolation conditions. *Mathematical Programming*, 100(1):183–215.

[259] Powell, M. J. (2015). On fast trust region methods for quadratic models with linear constraints. *Mathematical Programming Computation*, 7:237–267.

[260] Powell, M. J. et al. (2009). The bobyqa algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 26.

[261] Powell, M. J. D. (1994). *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pages 51–67. Springer Netherlands, Dordrecht.

[262] Powell, M. J. D. (1998). Direct search algorithms for optimization calculations. *Acta Numerica*, 7:287–336.

[263] Prékopa, A. (1970). On Probabilistic Constrained Programming. In *Proceedings of the Princeton Symposium on Mathematical Programming*, Princeton, NJ. Princeton University Press.

[264] Qian, E., Peherstorfer, B., O'Malley, D., Vesselinov, V. V., and Willcox, K. (2018). Multifidelity monte carlo estimation of variance and sensitivity indices. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):683–706.

[265] Quan, N. N. H., Van Lam, P., et al. (2021). Wind turbine blade design optimization using openfoam and dakota software. *Transportation Research Procedia*, 56:71–78.

[266] Quick, J., Annoni, J., King, R., Dykes, K., Fleming, P., and Ning, A. (2017). Optimization under uncertainty for wake steering strategies. *Journal of Physics: Conference Series*, 854:012036.

[267] Quick, J., King, J., King, R. N., Hamlington, P. E., and Dykes, K. (2020). Wake steering optimization under uncertainty. *Wind Energy Science*, 5(1):413–426.

[268] Quinonero-Candela, J. and Rasmussen, C. E. (2005). A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6:1939–1959.

[269] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

[270] Regis, R. G. (2011). Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers & Operations Research*, 38(5):837–853.

[271] Regis, R. G. (2014). Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2):218–243.

[272] Regis, R. G. and Wild, S. M. (2017). Conorbit: constrained optimization by radial basis function interpolation in trust regions. *Optimization Methods and Software*, 32(3):552–580.

[273] Rinaldi, F., Vicente, L. N., and Zeffiro, D. (2023). Stochastic trust-region and direct-search methods: A weak tail bound condition and reduced sample sizing.

[274] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407.

[275] Rockafellar, R. T., Uryasev, S., and Zabarankin, M. (2002). Deviation measures in risk analysis and optimization. Technical report, Research Report 2002-7, Risk Management and Financial Engineering Lab, Center for Applied Optimization, University of Florida.

[276] Rothfuss, P. (2011). *The Wise Man's Fear*. Kingkiller Chronicle. Astra Publishing House.

[277] Rubinstein, R. Y. and Shapiro, A. (1993). *Discrete event systems*. John Wiley & Sons Chichester New York.

[278] Sampaio, P. R. and Toint, P. L. (2015). A derivative-free trust-funnel method for equality-constrained nonlinear optimization. *Compuational Optimization and Applications*, 61(1):25–49.

[279] Sanderson, B. (2017). *Oathbringer*. Stormlight archive. Tor Fantasy.

[280] Scheuerer, M., Schaback, R., and Schlather, M. (2013). Interpolation of spatial data - A stochastic or a deterministic problem? *European Journal of Applied Mathematics*, 24(04):601–629.

[281] Schittkowski, K. (1987). *More Test Examples for Nonlinear Programming Codes*. Springer Berlin Heidelberg.

[282] Schittkowski, K. (2008). An Updated Set of 306 Test Problems for Nonlinear Programming with Validated Optimal Solutions -User's Guide. Technical report, University of Bayreuth.

[283] ScienceDaily. Optimization in sport optimal fixture scheduling. `https://www.sciencedaily.com/releases/2016/03/160310112053.html`, [Online, accessed 05-June-2023].

[284] Seeger, M., Williams, C. K. I., and Lawrence, N. D. (2003). Fast forward selection to speed up sparse gaussian process regression. In *In Workshop on AI and Statistics 9*.

[285] Shapiro, A., Dentcheva, D., and Ruszczynski, A. (2009). *Lectures on stochastic programming*. Society for Industrial and Applied Mathematics and the Mathematical Programming Society.

[286] Shashaani, S., Fatemeh, H., and Raghu, P. (2015). ASTRO-DF: a class of adaptive sampling trust-region algorithms for derivative-free simulation optimization. *Optimization online*.

[287] Shields, M. D., Teferra, K., Hapij, A., and Daddazio, R. P. (2015). Refined stratified sampling for efficient monte carlo based uncertainty quantification. *Reliability Engineering & System Safety*, 142:310–325.

[288] Silva, C. J. and Torres, D. F. M. (2006). Two-dimensional newton's problem of minimal resistance.

[289] Simionescu, P. and Beale, D. (2002). New concepts in graphic visualization of objective functions. volume 2.

[290] Smith, R. (2013). *Uncertainty Quantification: Theory, Implementation, and Applications*. Computational Science and Engineering. Society for Industrial and Applied Mathematics.

[291] Smola, A. J. and Bartlett, P. L. (2001). Sparse Greedy Gaussian Process Regression. In Keen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press.

[292] Smola, A. J. and Schökopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 911–918, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[293] Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian Processes using Pseudo-inputs. *Advances in Neural Information Processing Systems 18*, pages 1257–1264.

[294] Sousedík, B. and Elman, H. C. (2016). Stochastic galerkin methods for the steady-state navier–stokes equations. *Journal of Computational Physics*, 316:435–452.

[295] Spall, J. C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341.

[296] Spall, J. C. (1998). Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3):817–823.

[297] Spendley, W., Hext, G. R., and Himsworth, F. R. (1962). Sequential application of simplex design in optimisation and evolutionary operation. *Technometrics*, 4:441–461.

[298] Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

[299] Stuart, A. and Teckentrup, A. (2016). Posterior consistency for gaussian process approximations of bayesian posterior distributions. *Mathematics of Computation*, 87.

[300] Stubbs, R. A. and Mehrotra, S. (1999). A branch-and-cut method for 0-1 mixed convex programming. *Mathematical programming*, 86(3):515–532.

[301] Sullivan, T. J. (2015). *Stochastic Galerkin Methods*, pages 251–276. Springer International Publishing, Cham.

[302] Szegö, G. (2002). Measure of risk. *Journal of Banking & Finance*, 26:1253–1272.

[303] Tikhomirov, V. M. (1990). *Stories about maxima and minima.* American Mathematical Society.

[304] Tolkien, J. (2001a). *The Lord of the Rings: The Fellowship of the Ring.* Collins modern classics. Harper Collins.

[305] Tolkien, J. (2001b). *The Lord of the Rings: The Return of the King.* Collins modern classics. Harper Collins.

[306] Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on optimization*, 7(1):1–25.

[307] Tseng, P. (1999). Fortified-descent simplicial search method: A general approach. *SIAM Journal on Optimization*, 10(1):269–288.

[308] Uhlenbeck, G. E. and Ornstein, L. S. (1930). On the theory of the brownian motion. *Phys. Rev.*, 36:823–841.

[309] Uryasev, S. (1995). Derivatives of probability functions and some applications. *Annals of Operations Research*, 56:287–311.

[310] Uryasev, S. P. (2000). *Probabilistic Constrained Optimization: Methodology and Applications*, chapter Introduction to the theory of probabilistic functions and percentiles, pages 1–25. Kluwer Academic Publishers.

[311] Vanhatalo, J. and Vehtari, A. (2008). Modelling local and global phenomena with sparse gaussian processes. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI'08, pages 571–578, Arlington, Virginia, United States. AUAI Press.

[312] Venturi, L., Ballarin, F., and Rozza, G. (2019). A Weighted POD Method for Elliptic PDEs with Random Inputs. *Journal of Scientific Computing*, 81(1):136–153.

[313] Wahba, G. (1990). 7. Finite-Dimensional Approximating Subspaces. In *Spline Models for Observational Data*, pages 95–99. Society for Industrial and Applied Mathematics.

[314] Walder, C., Kim, K. I., and Schölkopf, B. (2008). Sparse multiscale gaussian process regression. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1112–1119, New York, NY, USA. ACM.

[315] Wang, I.-J. and Spall, J. C. (2003). Stochastic optimization with inequality constraints using simultaneous perturbation and penalty functions. In *Proceedings of the 42nd IEEE, Conference on decision and control.*

[316] Weirs, V. G. (2014). Dakota uncertainty quantification methods applied to the NEK-5000 SAHEX model. Technical report, United States.

[317] Weisburd, D., Britt, C., Wilson, D. B., and Wooditch, A. (2020). *Measuring Association for Scaled Data: Pearson's Correlation Coefficient*, pages 479–530. Springer International Publishing, Cham.

[318] Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D. (1992). Screening, predicting, and computer experiments. *Technometrics*, 34(1):15–25.

[319] Wendland, H. (2004). *Scattered Data Approximation.* Cambridge University Press, Cambridge.

[320] Wiener, N. (1938). The homogeneous chaos. *American Journal of Mathematics*, 60(4):897–936.

[321] Wild, S. M., Regis, R. G., and Shoemaker, C. A. (2008). ORBIT: optimization by radial basis function interpolation in trust-regions. *SIAM Journal on Scientific Computing*, 30(6):3197–3219.

[322] Wild, S. M. and Shoemaker, C. (2011). Global convergence of radial basis function trust region derivative-free algorithms. *SIAM Journal on Optimization*, 21(3):761–781.

[323] Williams, C., Rasmussen, C., Schwaighofer, A., and Tresp, V. (2002). Observations on the nyström method for gaussian process prediction. Technical report, Max Planck Institute for Biological Cybernetics, Tübingen, Germany.

[324] Williams, C. K. I. and Seeger, M. (2000). The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1159–1166. Morgan Kaufmann.

[325] Wilson, A. G., Dann, C., and Nickisch, H. (2015). Thoughts on massively scalable gaussian processes.

[326] Winfield, D. (1973). Function minimization by interpolation in a data table. *IMA Journal of Applied Mathematics*, 12(3):339–347.

[327] Winfield, D. H. (1970). *Function and functional optimization by interpolation in data tables.* PhD thesis, Harvard University.

[328] Witteveen, J. A. and Bijl, H. (2012). *Modeling Arbitrary Uncertainties Using Gram-Schmidt Polynomial Chaos.*

[329] Xiu, D. (2010). *Numerical Methods for Stochastic Computations—A Spectral Method Approach.* Princeton University Press, Princeton.

[330] Xiu, D. (2016). *Stochastic Collocation Methods: A Survey*, pages 1–18. Springer International Publishing, Cham.

[331] Xiu, D. and Karniadakis, G. E. (2002). The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, 24(2):619–644.

[332] Xiu, D. and Shen, J. (2009). Efficient stochastic galerkin methods for random diffusion equations. *Journal of Computational Physics*, 228(2):266–281.

[333] Yang, X.-S. (2010). *Engineering optimization: an introduction with metaheuristic applications.* John Wiley & Sons.

[334] Yuan, Y.-X. (2015). Recent advances in trust region algorithms. *Mathematical Programming*, 151(1):249–281.

[335] Zeng, K., Hou, J., Ivanov, K., and Jessee, M. A. (2019). Uncertainty quantification and propagation of multiphysics simulation of the pressurized water reactor core. *Nuclear Technology*, 205(12):1618–1637.

[336] Zhang, L. (2007). Sample mean and sample variance. *The American Statistician*, 61(2):159–160.

[337] Zieliński, R. (1988). A distribution-free median-unbiased quantile estimator. *Statistics*, 19(2):223–227.

[338] Zieliński, R. (2004). Optimal quantile estimators; small sample approach. Technical report, IMPAN, preprint 653.

[339] Zieliński, R. (2009). Optimal nonparametric quantile estimators. towards a general theory. a survey. *Communications in Statistics - Theory and Methods*, 38:980–992.

[340] Ørsted (2018). Horns Rev 2 Offshore Wind Farm. `https://orsted.com/-/media/www/docs/corp/com/our-business/wind-power/wind-farm-project-summary/horns-rev-2_uk_2018.ashx?la=en&hash=c2efe01aac0b9706fc81774e5b8c7a3c`. [Online; accessed 08-August-2023].

Part VII

Appendix

"Well, you can go on looking forward," said Gandalf. "There may be many unexpected feasts ahead of you."

—J.R.R. Tolkien [304]

A Multilevel methods

A.1 Proof for optimal resource allocation for multilevel Monte Carlo mean estimator

Referenced in Lemma 6.

Proof. The optimization problem is given as:

$$\begin{aligned} \overset{*}{N}{}_\ell^{\mathbb{E}} = \arg\min_{N_\ell^{\mathbb{E}}} C_T^{\mathbb{E}} &:= \sum_{\ell=1}^{L} C_\ell N_\ell^{\mathbb{E}}, \\ \text{s.t. } \mathbb{V}[\widehat{\mu}_{1,\text{ML}}] &= \epsilon_{\mathbb{E}}^2, \end{aligned} \tag{A.1}$$

where $\mathbb{V}[\widehat{\mu}_{1,\text{ML}}] = \sum_{\ell=1}^{L} \frac{1}{N_\ell^{\mathbb{E}}} \mathbb{V}[Q_\ell - Q_{\ell-1}]$.

We employ the method of Lagrange multiplier for equality constraint problems and form the Lagrange function

$$L(N_\ell^{\mathbb{E}}, \lambda) = \sum_{\ell=1}^{L} C_\ell N_\ell^{\mathbb{E}} + \lambda^2 (\sum_{\ell=1}^{L} \frac{1}{N_\ell^{\mathbb{E}}} \widehat{\mu}_2[Q_\ell - Q_{\ell-1}] - \epsilon_{\mathbb{E}}^2). \tag{A.2}$$

Next, we compute the partial derivatives for $\{N_\ell^{\mathbb{E}}\}_{i=1}^{L}$ and λ and set them to 0

$$\begin{aligned} \frac{\partial L}{\partial N_1^{\mathbb{E}}} &= C_1 - \lambda^2 \frac{1}{(N_1^{\mathbb{E}})^2} \widehat{\mu}_2[Q_1 - Q_0] \overset{!}{=} 0 \\ &\vdots \\ \frac{\partial L}{\partial N_L^{\mathbb{E}}} &= C_L - \lambda^2 \frac{1}{(N_L^{\mathbb{E}})^2} \widehat{\mu}_2[Q_L - Q_{L-1}] \overset{!}{=} 0 \\ \frac{\partial L}{\partial \lambda^2} &= \sum_{\ell=1}^{L} \frac{1}{N_\ell^{\mathbb{E}}} \widehat{\mu}_2[Q_\ell - Q_{\ell-1}] - \epsilon_{\mathbb{E}}^2 \overset{!}{=} 0. \end{aligned} \tag{A.3}$$

For $\frac{\partial L}{\partial N_\ell^{\mathbb{E}}}$, we get $N_\ell^{\mathbb{E}} = \lambda \sqrt{\frac{\widehat{\mu}_2[Q_\ell - Q_{\ell-1}]}{C_\ell}}$.

We plug this into the last equation:

$$\begin{aligned}\sum_{\ell=1}^{L} \frac{1}{N_\ell^{\mathbb{E}}} \widehat{\mu}_2[Q_\ell - Q_{\ell-1}] - \epsilon_{\mathbb{E}}^2 &= 0 \\ \Leftrightarrow \frac{1}{\lambda} \sum_{\ell=1}^{L} \sqrt{\frac{C_\ell}{\widehat{\mu}_2[Q_\ell - Q_{\ell-1}]}} \widehat{\mu}_2[Q_\ell - Q_{\ell-1}] - \epsilon_{\mathbb{E}}^2 &= 0 \\ \Leftrightarrow \frac{1}{\lambda} \sum_{\ell=1}^{L} \sqrt{C_\ell \widehat{\mu}_2[Q_\ell - Q_{\ell-1}]} &= \epsilon_{\mathbb{E}}^2 \\ \Leftrightarrow \lambda = \epsilon_{\mathbb{E}}^{-2} \sum_{\ell=1}^{L} \sqrt{C_\ell \widehat{\mu}_2[Q_\ell - Q_{\ell-1}]}. \end{aligned} \tag{A.4}$$

□

B Gaussian processes

Most of the following properties, identities and equation can be found in [254] and [141].

B.1 Matrix properties

B.1.1 Woodbury identity

Referenced in Section 17.2 and Eq. (17.33).

We find different version of the Woodbury identity in literature [254]:

$$(\mathbf{A} + \mathbf{C}\mathbf{B}\mathbf{C}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{C}(\mathbf{B}^{-1} + \mathbf{C}^T\mathbf{A}^{-1}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{A}^{-1} \tag{B.1}$$

$$(\mathbf{A} + \mathbf{U}\mathbf{B}\mathbf{V}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{B}^{-1} + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^T\mathbf{A}^{-1} \tag{B.2}$$

B.1.2 Matrix derivatives and determinant

We decribe different rules regarding the derivative of a matrix and its determinant [254]:

$$\partial(\alpha\mathbf{X}) = \alpha(\partial\mathbf{X}), \alpha \in \mathbb{R} \tag{B.3}$$

$$\partial\text{Tr}(\mathbf{X}) = \text{Tr}(\partial\mathbf{X}) \tag{B.4}$$

$$\partial\mathbf{X}^{-1} = \mathbf{X}^{-1}(\partial\mathbf{X})\mathbf{X}^{-1} \tag{B.5}$$

$$\partial|\mathbf{X}| = |\mathbf{X}|\text{Tr}(\mathbf{X}^{-1}\partial\mathbf{X}) \tag{B.6}$$

$$\partial(\log|\mathbf{X}|) = \text{Tr}(\mathbf{X}^{-1}\partial\mathbf{X}) \tag{B.7}$$

$$\partial\mathbf{X}^T = (\partial\mathbf{X})^T \tag{B.8}$$

B.2 Drawing functions from covariance kernel

Referenced in Section 7.1.

A covariance function represents a distribution over functions and employing it, we can draw samples from this function space. The following steps show how to sample a function by using the mean and covariance function:

Goal: Sample $\mathbf{y} \sim \mathcal{N}(\mathbf{y}|\mathbf{m}, \mathbf{K}(\mathbf{X}, \mathbf{X}))$.

1 Compute $\mathbf{K}(\mathbf{X}, \mathbf{X})$ for sample location $\mathbf{x}_i, i = 1, ..., n$.

2.a Cholesky decomposition: $\mathbf{K} = \mathbf{L}\mathbf{L}^T$.

2.b LDL decomposition: $\mathbf{K} = \mathbf{L}'\mathbf{D}\mathbf{L}'^T = \mathbf{L}'\mathbf{D}^{\frac{1}{2}}(\mathbf{L}'\mathbf{D}^{\frac{1}{2}})^T = \mathbf{L}\mathbf{L}^T$.

3. Generate: $\mathbf{u} \sim \mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{I})$.
4. Compute: $\mathbf{y} = \mathbf{m} + \mathbf{L}\mathbf{u}$.

B.3 Gaussian distribution properties for marginal distributions

Referenced in Section 7.2.

Given a marginal Gaussian distribution for $\mathbf{x}$

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mu, \mathbf{\Lambda}^{-1}), \tag{B.9}$$

and a conditional Gaussian distribution for $\mathbf{y}$ conditioned on $\mathbf{x}$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}). \tag{B.10}$$

Then the marginal distribution of $\mathbf{y}$ is given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mu + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\mathbf{\Lambda}^{-1}\mathbf{A}^T), \tag{B.11}$$

and conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ is given by

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|\mathbf{A}\mu + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\mathbf{\Lambda}^{-1}\mathbf{A}^T), \\ p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x}|\mathbf{\Sigma}(\mathbf{A}^T\mathbf{L}(\mathbf{y} - \mathbf{b}) + \mathbf{\Lambda}\mu), \mathbf{\Sigma}), \end{aligned} \tag{B.12}$$

with

$$\mathbf{\Sigma} = (\mathbf{\Lambda} + \mathbf{A}^T\mathbf{L}\mathbf{A})^{-1}. \tag{B.13}$$

Further details can also be found in [47] and [269].

B.4 Conditional distribution of multivariate gaussians

Referenced in Section 7.2.

Lemma 23. *Given a multivariate normal vector* $\mathbf{y} \sim \mathcal{N}(\mathbf{y}|\mu, \mathbf{\Sigma})$, *which can be partitioned as*

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \tag{B.14}$$

and similarly

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12} \\ \mathbf{\Sigma}_{21} & \mathbf{\Sigma}_{22} \end{bmatrix}. \tag{B.15}$$

The conditioned distribution of $\mathbf{y}_2$ *conditioned on* $\mathbf{y}_1$ *is given as*

$$\mathbf{y}_2|\mathbf{y}_1 \sim \mathcal{N}(\mathbf{y}_2|\overline{\mu}, \overline{\mathbf{\Sigma}}) \tag{B.16}$$

where

$$\begin{aligned} \overline{\mu} &= \mu_2 + \mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}(\mathbf{y}_1 - \mu_1), \\ \overline{\mathbf{\Sigma}} &= \mathbf{\Sigma}_{22} - \mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}\mathbf{\Sigma}_{12}. \end{aligned} \tag{B.17}$$

Proof. Specific properties for multivariate Gaussians, which are used in the following are described in [47, 269, 254].

We define $\mathbf{z} := \mathbf{y}_2 + \mathbf{A}\mathbf{y}_1$, where $\mathbf{A} = -\mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}$ and write

$$\begin{aligned}\mathbb{C}\mathrm{ov}[(,]z, \mathbf{y}_1) &= \mathbb{C}\mathrm{ov}[(,]y_2, \mathbf{y}_1) + \mathbb{C}\mathrm{ov}[(,]A\mathbf{y}_1, \mathbf{y}_1) \\ &= \mathbf{\Sigma}_{21} + \mathbf{A}\mathbb{V}[\mathbf{y}_1] \\ &= \mathbf{\Sigma}_{21} - \mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}\mathbf{\Sigma}_{11} \\ &= 0.\end{aligned} \tag{B.18}$$

This implies that $\mathbf{z}$ and $\mathbf{y}_1$ are uncorrelated and, thus, independent. Employing the linearity of the expected value, we derive the expected value of $\mathbf{z}$ as

$$\begin{aligned}\mathbb{E}[\mathbf{z}] &= \mathbb{E}[\mathbf{y}_2 + \mathbf{A}\mathbf{y}_1] \\ &= \mathbb{E}[\mathbf{y}_2] + \mathbf{A}\mathbb{E}[\mathbf{y}_1] \\ &= \mu_2 + \mathbf{A}\mu_1,\end{aligned} \tag{B.19}$$

and it follows

$$\begin{aligned}\mathbb{E}[\mathbf{y}_2|\mathbf{y}_1] &= \mathbb{E}[\mathbf{z} - \mathbf{A}\mathbf{y}_1|\mathbf{y}_1] \\ &= \mathbb{E}[\mathbf{z}|\mathbf{y}_1] - \mathbb{E}[\mathbf{A}\mathbf{y}_1|\mathbf{y}_1] \\ &= \mathbb{E}[z] - \mathbf{A}\mathbf{y}_1 \\ &= \mu_2 + \mathbf{A}\mu_1 - \mathbf{A}\mathbf{y}_1 \\ &= \mu_2 + \mathbf{A}(\mu_1 - \mathbf{y}_1) \\ &= \mu_2 + \mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}(\mathbf{y}_1 - \mu_1).\end{aligned} \tag{B.20}$$

We assume a mean of 0 without loss of generality.

We have the following relationship for the covariance matrix

$$\begin{aligned}\mathbb{V}[\mathbf{y}_2|\mathbf{y}_1] &= \mathbb{V}[\mathbf{z} - \mathbf{A}\mathbf{y}_1|\mathbf{y}_1] \\ &= \mathbb{V}[\mathbf{z}|\mathbf{y}_1] + \mathbb{V}[\mathbf{A}\mathbf{y}_1|\mathbf{y}_1] - \mathbf{A}\mathbb{C}\mathrm{ov}[(,]z, -\mathbf{y}_1) - \mathbb{C}\mathrm{ov}[(,]z, -\mathbf{y}_1)\mathbf{A}^T \\ &= \mathbb{V}[\mathbf{z}|\mathbf{y}_1] \\ &= \mathbb{V}[\mathbf{z}].\end{aligned} \tag{B.21}$$

We prove the conditional variance employing above properties as

$$\begin{aligned}\mathbb{V}[\mathbf{y}_2|\mathbf{y}_1] = \mathbb{V}[\mathbf{z}] = \mathbb{V}[\mathbf{y}_2 + \mathbf{A}\mathbf{y}_1] &= \\ &= \mathbb{V}[\mathbf{y}_2] + \mathbf{A}\mathbb{V}[\mathbf{y}_1]\mathbf{A}^T + \mathbf{A}\mathbb{C}\mathrm{ov}[(,]y_2, \mathbf{y}_1) + \mathbb{C}\mathrm{ov}[(,]y_1, \mathbf{y}_2)\mathbf{A}^T \\ &= \mathbf{\Sigma}_{22} + \mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}\mathbf{\Sigma}_{11}\mathbf{\Sigma}_{11}^{-1}\mathbf{\Sigma}_{12} - 2\mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}\mathbf{\Sigma}_{12} \\ &= \mathbf{\Sigma}_{22} + \mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}\mathbf{\Sigma}_{12} - 2\mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}\mathbf{\Sigma}_{12} \\ &= \mathbf{\Sigma}_{22} - \mathbf{\Sigma}_{21}\mathbf{\Sigma}_{11}^{-1}\mathbf{\Sigma}_{12}.\end{aligned} \tag{B.22}$$

□

B.5 Marginal likelihood optimization

B.5.1 Derivatives of log marginal likelihood terms

Referenced in Section 17.2.

We present the derivatives of different terms of the log marginal likelihood for aGP. We limit ourselves to FITC though these results similarly extend to SOR and DTC. We start off defining the different terms:

$$\begin{aligned} L_1 &:= \log|\mathbf{\Lambda} + {\sigma_{\mathbf{n}}}^2\mathbf{I}|, \\ L_2 &:= \log|\mathbf{K_{UU}}|, \\ L_3 &:= \log|\mathbf{K_{UU}} + \mathbf{K_{UX}}(\mathbf{\Lambda} + {\sigma_{\mathbf{n}}}^2\mathbf{I})^{-1}\mathbf{K_{XU}}|, \\ L_4 &:= \mathbf{y}^T\mathbf{\Gamma}^{-1}\mathbf{y}, \end{aligned} \tag{B.23}$$

where $\mathbf{\Gamma} = (\mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\mathbf{K_{UX}} + \mathbf{\Lambda} + {\sigma_{\mathbf{n}}}^2\mathbf{I})$ is the noise matrix. The gradient is computed with respect to the hyperparamter vector $\boldsymbol{\psi}_{\text{aGP}} = [\sigma_n^f, l_1, ..., l_D, \mathbf{U}]$. A matrix derivative is abbreviated as $\dot{A} := \frac{\partial A}{\partial \boldsymbol{\psi}_{\text{AGP}}}$ for convenience.

The derivative of L_1, where $\mathbf{\Lambda} = \text{diag}(\mathbf{K_{XX}} - \mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\mathbf{K_{UX}})$ is given as:

$$\begin{aligned} \frac{\partial L_1}{\partial \boldsymbol{\psi}_{\text{AGP}}} &= \frac{\partial}{\partial \boldsymbol{\psi}_{\text{AGP}}}(\log|\mathbf{\Lambda} + {\sigma_{\mathbf{n}}}^2\mathbf{I}|) \\ &= \text{Tr}\Big((\mathbf{\Lambda} + {\sigma_{\mathbf{n}}}^2\mathbf{I})^{-1}\dot{\mathbf{\Lambda}}\Big) \\ &= \text{Tr}\Big((\mathbf{\Lambda} + {\sigma_{\mathbf{n}}}^2\mathbf{I})^{-1} \\ &\quad \underbrace{\text{diag}(\dot{\mathbf{K}}_{\mathbf{XX}} - \dot{\mathbf{K}}_{\mathbf{XU}}\mathbf{K_{UU}^{-1}}\mathbf{K_{UX}} + \mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\dot{\mathbf{K}}_{\mathbf{UU}}\mathbf{K_{UU}^{-1}}\mathbf{K_{UX}} - \mathbf{K_{XU}}\mathbf{K_{UU}^{-1}}\dot{\mathbf{K}}_{\mathbf{UX}})}_{\dot{\mathbf{\Lambda}}}\Big). \end{aligned} \tag{B.24}$$

The derivative for L_2 is given as:

$$\begin{aligned} \frac{\partial L_2}{\partial \boldsymbol{\psi}_{\text{AGP}}} &= \frac{\partial}{\partial \boldsymbol{\psi}_{\text{AGP}}}(\log|\mathbf{K_{UU}}|) \\ &= \text{Tr}(\mathbf{K_{UU}^{-1}}\dot{\mathbf{K}}_{\mathbf{UU}}). \end{aligned} \tag{B.25}$$

The derivative for L_3 is given as:

$$\begin{aligned}
\frac{\partial L_3}{\partial \boldsymbol{\psi}_{\text{AGP}}} &= \frac{\partial}{\partial \boldsymbol{\psi}_{\text{AGP}}}(\log|\mathbf{K_{UU}} + \mathbf{K_{UX}}(\boldsymbol{\Lambda} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{K_{XU}}|) \\
&= \operatorname{Tr}\Big((\mathbf{K_{UU}} + \mathbf{K_{UX}}(\boldsymbol{\Lambda} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{K_{XU}})^{-1}\frac{\partial}{\partial \boldsymbol{\psi}_{\text{AGP}}}(\mathbf{K_{UU}} + \mathbf{K_{UX}}(\boldsymbol{\Lambda} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{K_{XU}})\Big) \\
&= \operatorname{Tr}\Big((\mathbf{K_{UU}} + \mathbf{K_{UX}}(\boldsymbol{\Lambda} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{K_{XU}})^{-1}[\dot{\mathbf{K}}_{\mathbf{UU}} + \frac{\partial}{\partial \boldsymbol{\psi}_{\text{AGP}}}(\mathbf{K_{UX}}(\boldsymbol{\Lambda} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{K_{XU}})]\Big) \\
&= \operatorname{Tr}\Big((\mathbf{K_{UU}} + \mathbf{K_{UX}}(\boldsymbol{\Lambda} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{K_{XU}})^{-1} \\
&\quad [\dot{\mathbf{K}}_{\mathbf{UU}} + \dot{\mathbf{K}}_{\mathbf{UX}}(\boldsymbol{\Lambda} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{K_{XU}} + \mathbf{K_{UX}}(\boldsymbol{\Lambda} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\dot{\mathbf{K}}_{\mathbf{XU}} \\
&\quad - \mathbf{K_{UX}}(\boldsymbol{\Lambda} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\dot{\boldsymbol{\Lambda}}(\boldsymbol{\Lambda} + \sigma_{\mathbf{n}}{}^2\mathbf{I})^{-1}\mathbf{K_{UX}}]\Big).
\end{aligned} \tag{B.26}$$

The derivative for L_4 is given as:

$$\begin{aligned}
\frac{\partial L_4}{\partial \boldsymbol{\psi}_{\text{AGP}}} &= \frac{\partial}{\partial \boldsymbol{\psi}_{\text{AGP}}}(\mathbf{y}^T\boldsymbol{\Gamma}^{-1}\mathbf{y}) \\
&= -\mathbf{y}^T\boldsymbol{\Gamma}^{-1}\dot{\boldsymbol{\Gamma}}\boldsymbol{\Gamma}^{-1}\mathbf{y} \\
&= -\mathbf{y}^T\boldsymbol{\Gamma}^{-1}\frac{\partial}{\partial \boldsymbol{\psi}_{\text{AGP}}}(\mathbf{K_{XU}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K_{UX}} + \boldsymbol{\Lambda} + \sigma_{\mathbf{n}}{}^2\mathbf{I})\boldsymbol{\Gamma}^{-1}\mathbf{y} \\
&= -\mathbf{y}^T\boldsymbol{\Gamma}^{-1}(\frac{\partial}{\partial \boldsymbol{\psi}_{\text{AGP}}}(\mathbf{K_{XU}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K_{UX}}) + \dot{\boldsymbol{\Lambda}})\boldsymbol{\Gamma}^{-1}\mathbf{y} \\
&= -\mathbf{y}^T\boldsymbol{\Gamma}^{-1} \\
&\quad (\dot{\mathbf{K}}_{\mathbf{XU}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K_{UX}} - \mathbf{K_{XU}}\mathbf{K}_{\mathbf{UU}}^{-1}\dot{\mathbf{K}}_{\mathbf{UU}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K_{UX}} + \mathbf{K_{XU}}\mathbf{K}_{\mathbf{UU}}^{-1}\dot{\mathbf{K}}_{\mathbf{UX}} + \dot{\boldsymbol{\Lambda}}) \\
&\quad \boldsymbol{\Gamma}^{-1}\mathbf{y}.
\end{aligned} \tag{B.27}$$

The derivatives of the different kernel matrices are given in App. B.5.2.

B.5.2 Derivatives of kernel matrices

Referenced in Section 17.2.

We present the derivatives of the kernel matrices $\mathbf{K_{XX}}, \mathbf{K_{XU}}, \mathbf{K_{UX}}, \mathbf{K_{UU}}$ in relation to the hyperparameters $\boldsymbol{\psi}_{\text{aGP}} = [\sigma_n^f, l_1, ..., l_D, \mathbf{U}]$. It is essential to acknowledge that these derivations are based on the application of the squared exponential kernel. This kernel can be expressed as a product over one dimension, a characteristic that we exploit throughout this chapter. The kernel is defined as:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \prod_{d=1}^{D} \exp\left(\frac{(x_i^{(d)} - x_j^{(d)})^2}{2l_d^{-1}}\right). \tag{B.28}$$

We compute the individual partial derivatives for σ_f^2 and $\mathbf{l}$ as:

$$\begin{aligned}
\frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial \sigma_f^2} &= \prod_{d=1}^{D} \exp\left(\frac{(x_i^{(d)} - x_j^{(d)})^2}{2l_d^{-1}}\right), \\
\frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial l_k} &= \sigma_f^2 \frac{(x_i^{(k)} - x_j^{(k)})^2}{2l_k^2} \prod_{d=1}^{D} \exp\left(\frac{(x_i^{(d)} - x_j^{(d)})^2}{2l_d^{-1}}\right).
\end{aligned} \tag{B.29}$$

This requires that $\mathbf{x}_i$ is an element of $\mathbf{X}$ or $\mathbf{U}$. . We then calculate the matrix derivatives $\frac{\partial \mathbf{K_{XX}}}{\partial \sigma_f^2}$, $\frac{\partial \mathbf{K_{XU}}}{\partial \sigma_f^2}$, $\frac{\partial \mathbf{K_{UX}}}{\partial \sigma_f^2}$, $\frac{\partial \mathbf{K_{XX}}}{\partial l_k}$, $\frac{\partial \mathbf{K_{XU}}}{\partial l_k}$, and $\frac{\partial \mathbf{K_{UX}}}{\partial l_k}$ by applying the derived partial derivatives to each constituent element.

The derivatives of the induced points are given as:

$$\begin{aligned}
\frac{\partial k(\mathbf{u}_i, \mathbf{u}_j)}{\partial u_i^{(k)}} &= -\frac{(u_i^{(k)} - u_j^{(k)})}{l_k} \prod_{d=1}^{D} \exp\left(\frac{(u_i^{(d)} - u_j^{(d)})^2}{2l_d^{-1}}\right), \\
\frac{\partial k(\mathbf{u}_i, \mathbf{u}_i)}{\partial u_i^{(k)}} &= 1, \\
\frac{\partial k(\mathbf{u}_i, \mathbf{u}_j)}{\partial u_h^{(k)}} &= 0, h \neq i, h \neq j, \\
\frac{\partial k(\mathbf{u}_i, \mathbf{x}_j)}{\partial u_i^{(k)}} &= -\frac{(u_i^{(k)} - x_j^{(k)})}{l_k} \prod_{d=1}^{D} \exp\left(\frac{(u_i^{(d)} - x_j^{(d)})^2}{2l_d^{-1}}\right).
\end{aligned} \tag{B.30}$$

The resulting matrices of different combinations of $\mathbf{X}$ and $\mathbf{U}$ are given as:

$$\begin{aligned}
\frac{\partial \mathbf{K_{UU}}}{\partial u_i^{(k)}} &= \begin{pmatrix} 0 & \cdots & \frac{\partial k(\mathbf{u}_1,\mathbf{u}_j)}{\partial u_i^{(k)}} & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ \frac{\partial k(\mathbf{u}_i,\mathbf{u}_1)}{\partial u_i^{(k)}} & \cdots & \frac{\partial k(\mathbf{u}_i,\mathbf{u}_i)}{\partial u_i^{(k)}} & \cdots & \frac{\partial k(\mathbf{u}_i,\mathbf{u}_M)}{\partial u_i^{(k)}} \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial k(\mathbf{u}_M,\mathbf{u}_j)}{\partial u_i^{(k)}} & \cdots & 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 & \cdots & \frac{\partial k(\mathbf{u}_1,\mathbf{u}_j)}{\partial u_i^{(k)}} & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ \frac{\partial k(\mathbf{u}_i,\mathbf{u}_1)}{\partial u_i^{(k)}} & \cdots & 1 & \cdots & \frac{\partial k(\mathbf{u}_i,\mathbf{u}_M)}{\partial u_i^{(k)}} \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial k(\mathbf{u}_M,\mathbf{u}_j)}{\partial u_i^{(k)}} & \cdots & 0 \end{pmatrix}, \\
\frac{\partial \mathbf{K_{UX}}}{\partial u_i^{(k)}} &= \begin{pmatrix} 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ \frac{\partial k(\mathbf{u}_i,\mathbf{u}_1)}{\partial u_i^{(k)}} & \cdots & \frac{\partial k(\mathbf{u}_i,\mathbf{u}_i)}{\partial u_i^{(k)}} & \cdots & \frac{\partial k(\mathbf{u}_i,\mathbf{u}_M)}{\partial u_i^{(k)}} \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{pmatrix}, \\
\frac{\partial \mathbf{K_{XU}}}{\partial u_i^{(k)}} &= (\frac{\partial \mathbf{K_{UX}}}{\partial u_i^{(k)}})^T
\end{aligned} \tag{B.31}$$

C Multilevel Monte Carlo for higher order moments

C.1 Proof: Unbiased estimator for variance

Referenced in Lemma 7.

Proof.

$$\begin{aligned}
\mathbb{E}[\widehat{\mu}_2] &= \mathbb{E}\left[\frac{1}{N-1}\sum_{i=1}^{N}(Q^{(i)}-\widehat{\mu}_1)^2\right] = \frac{1}{N-1}\sum_{i=1}^{N}\mathbb{E}\left[(Q^{(i)})^2 - 2Q^{(i)}\widehat{\mu}_1 + \widehat{\mu}_1^2\right] \\
&= \frac{1}{N-1}\sum_{i=1}^{N}\mathbb{E}\left[(Q^{(i)})^2\right] - \mathbb{E}\left[2Q^{(i)}\widehat{\mu}_1\right] + \mathbb{E}\left[\widehat{\mu}_1^2\right] \\
&= \frac{1}{N-1}\sum_{i=1}^{N}\mathbb{V}\left[Q^{(i)}\right] + \mathbb{E}\left[Q^{(i)}\right]^2 - 2\mathbb{E}\left[Q^{(i)}\frac{1}{N}\sum_{j=1}^{N}Q^{(j)}\right] \\
&\qquad + \mathbb{E}\left[\frac{1}{N}\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}Q^{(i)}Q^{(j)}\right] \\
&= \frac{1}{N-1}\sum_{i=1}^{N}\mathbb{V}[Q] + \mathbb{E}[Q]^2 - 2\frac{1}{N}\left(\mathbb{E}[Q^2] + (N-1)\mathbb{E}[Q]^2\right) \qquad \text{(C.1)} \\
&\qquad + \frac{1}{N}(\mathbb{E}[Q^2] + (N-1)\mathbb{E}[Q]^2) \\
&= \frac{1}{N-1}\sum_{i=1}^{N}\mathbb{V}[Q] + \mathbb{E}[Q]^2 - \frac{1}{N}\mathbb{E}[Q^2] - \frac{N-1}{N}\mathbb{E}[Q]^2 \\
&= \frac{1}{N-1}\sum_{i=1}^{N}\mathbb{V}[Q] + \frac{1}{N}\mathbb{E}[Q]^2 - \frac{1}{N}\mathbb{E}[Q^2] \\
&= \frac{1}{N-1}\sum_{i=1}^{N}\mathbb{V}[Q] - \frac{1}{N}\mathbb{V}[Q] = \frac{1}{N-1}\sum_{i=1}^{N}\frac{N-1}{N}\mathbb{V}[Q] \\
&= \mathbb{V}[Q].
\end{aligned}$$

□

C.2 Proof: Unbiased estimator for variance of variance

Referenced in Lemma 8.

Proof.

$$\begin{aligned}
\mathbb{E}\left[\frac{(N-1)}{N^2-2N+3}\left(\widehat{\mu}_4-\frac{N-3}{N-1}\widehat{\mu}_2^2\right)\right] &= \frac{(N-1)}{N^2-2N+3}\left(\mathbb{E}[\widehat{\mu}_4]-\frac{(N-3)}{(N-1)}\mathbb{E}[\widehat{\mu}_2^2]\right)\\
&= \frac{(N-1)}{N^2-2N+3}\left(\mu_4-\frac{N-3}{N-1}\mathbb{E}[\widehat{\mu}_2^2]\right)\\
&= \frac{(N-1)}{N^2-2N+3}\left(\mu_4-\frac{N-3}{N-1}\left[\frac{1}{N}\left(\mu_4-\frac{N-3}{N-1}\mu_2^2\right)+\mu_2^2\right]\right)\\
&= \frac{(N-1)}{N^2-2N+3}\left(\mu_4-\frac{N-3}{N(N-1)}\mu_4+\frac{(N-3)^2}{N(N-1)^2}\mu_2^2-\frac{N-3}{N-1}\mu_2^2\right)\\
&= \frac{(N-1)}{N^2-2N+3}\left[\left(1-\frac{N-3}{N(N-1)}\right)\mu_4-\left(1-\frac{(N-3)}{N(N-1)}\right)\frac{(N-3)}{(N-1)}\mu_2^2\right]\\
&= \frac{(N-1)}{N^2-2N+3}\left(1-\frac{N-3}{N(N-1)}\right)\frac{1}{N}\left(\mu_4-\frac{(N-3)}{(N-1)}\mu_2^2\right)\\
&= \frac{(N-1)}{N^2-2N+3}\left(\frac{N^2-2N+3}{N(N-1)}\right)\left(\mu_4-\frac{(N-3)}{(N-1)}\mu_2^2\right)\\
&= \frac{1}{N}\left(\mu_4-\frac{(N-3)}{(N-1)}\mu_2^2\right)
\end{aligned} \tag{C.2}$$

□

C.3 Proof: Unbiased estimator for fourth central moment

Referenced in Lemma 9.

Proof. A biased estimator for the fourth central moment is given in [101, p.268, after eq. (6)] as

$$\begin{aligned}\mathbb{E}[\widehat{\mu}_{4,\text{biased}}] &= \frac{(N-1)(N^2-3N+3)}{N^3}\mu_4 + \frac{3(2N-3)(N-1)}{N^3}\mu_2^2 \\ \Leftrightarrow \mu_4 &= \frac{1}{N^2-3N+3}\left(\frac{N^3}{N-1}\mathbb{E}[\widehat{\mu}_{4,\text{biased}}] - (6N-9)\mu_2^2\right).\end{aligned} \tag{C.3}$$

Note that this is an unbiased estimator only if we use the exact value for μ_2 since μ_2^2 is unbiased while $\widehat{\mu}_2^2$ is not. Therefore, we need an unbiased estimator for $\widehat{\mu}_2^2$ and we know that

$$\begin{aligned}\mathbb{E}[\widehat{\mu}_2^2] &= \mathbb{V}[\widehat{\mu}_2] + \mathbb{E}[\widehat{\mu}_2]^2 \\ &= \frac{1}{N}\left(\mu_4 - \frac{N-3}{N-1}\mu_2^2\right) + \mu_2^2.\end{aligned} \tag{C.4}$$

Using both (C.3) and (C.4) we get the result

$$\begin{aligned}&\mathbb{E}\left[\underbrace{\frac{1}{(N^2-3N+3) - \frac{(6N-9)(N^2-N)}{N(N^2-2N+3)}}}_{(*)}\left(\frac{N^3}{N-1}\widehat{\mu}_{4,\text{biased}} - \frac{(6N-9)(N^2-N)}{N^2-2N+3}\widehat{\mu}_2^2\right)\right] \\ &= (*)\left(\frac{N^3}{N-1}\mathbb{E}[\widehat{\mu}_{4,\text{biased}}] - \frac{(6N-9)(N^2-N)}{N^2-2N+3}\mathbb{E}[\widehat{\mu}_2^2]\right) \\ &= (*)\left(\frac{N^3}{N-1}\mathbb{E}[\widehat{\mu}_{4,\text{biased}}] - \frac{(6N-9)(N^2-N)}{N^2-2N+3}\left[\frac{1}{N}(\mu_4 - \frac{N-3}{N-1}\mu_2^2) + \mu_2^2\right]\right) \\ &= (*)\left(\frac{N^3}{N-1}\left[\frac{(N-1)(N^2-3N+3)}{N^3}\mu_4 + \frac{3(2N-3)(N-1)}{N^3}\mu_2^2\right] - \right. \\ &\qquad \left.\frac{(6N-9)(N^2-N)}{N^2-2N+3}\left[\frac{1}{N}(\mu_4 - \frac{N-3}{N-1}\mu_2^2) + \mu_2^2\right]\right) \\ &= \mu_4.\end{aligned} \tag{C.5}$$

□

C.4 Proof: Delta method

Referenced in Lemma 10.

Proof. We can find an approximation by using a Taylor expansion of $g(X)$ around μ_1, s.t.

$$g(X) = g(\mu_1) + g'(\mu_1)(X - \mu_1) + g''(\mu_1)\frac{(X - \mu_1)^2}{2!} + ..., \tag{C.6}$$

where we drop the higher terms to get

$$g(X) \approx g(\mu_1) + g'(\mu_1)(X - \mu_1). \tag{C.7}$$

Taking the variance on both sides yields

$$\mathbb{V}[g(X)] \approx \mathbb{V}[g(\mu_1)] + \mathbb{V}[g'(\mu_1)(X - \mu_1)] = g'(X)^2\mathbb{V}[X]. \tag{C.8}$$

Denoting $X := \widehat{\mu}_2$ and $g(X) := \sqrt{X}$, we get

$$\mathbb{V}\left[\sqrt{\widehat{\mu}_2}\right] \approx \left(\frac{1}{2\sqrt{\widehat{\mu}_2}}\right)^2 \mathbb{V}[\widehat{\mu}_2] = \frac{1}{4\widehat{\mu}_2}\mathbb{V}[\widehat{\mu}_2]. \tag{C.9}$$

□

C.5 Proof: Covariance of variance

Referenced in Lemma 11.

Proof. Change to centered moments using $Z_\ell^{(i)} = Q_\ell^{(i)} - \widehat{\mu}_{1,\ell}$ and $Z_{\ell-1}^{(i)} = Q_{\ell-1}^{(i)} - \widehat{\mu}_{1,\ell-1}$:

$$\widehat{\mu}_{2,\ell} = \frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell} Z_\ell^{(i)2} - \frac{1}{N_\ell(N_\ell - 1)}\left(\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\right)^2 \tag{C.10}$$

$$\widehat{\mu}_{2,\ell-1} = \frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)\ 2} - \frac{1}{N_\ell(N_\ell - 1)}\left(\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}\right)^2 \tag{C.11}$$

Plug back in, and split up covariance

$$\begin{aligned}
\mathbb{C}\mathrm{ov}[\widehat{\mu}_{2,\ell}, \widehat{\mu}_{2,\ell-1}] = {} & \frac{1}{(N_{\ell-1})^2}\mathbb{C}\mathrm{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)2}, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)\ 2}\right] && \text{(c1)}\\
& - \frac{1}{N_\ell(N_{\ell-1})^2}\mathbb{C}\mathrm{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)2}, \left(\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}\right)^2\right] && \text{(c2)}\\
& - \frac{1}{N_\ell(N_{\ell-1})^2}\mathbb{C}\mathrm{ov}\left[\left(\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\right)^2, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)\ 2}\right] && \text{(c3)}\\
& + \frac{1}{N_\ell^2(N_{\ell-1})^2}\mathbb{C}\mathrm{ov}\left[\left(\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\right)^2, \left(\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}\right)^2\right]. && \text{(c4)}
\end{aligned}$$

Solve the four different terms independently:

$$\begin{aligned}
&\text{(c1) } \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}, \sum_{i=1}^{N_\ell} {Z_{\ell-1}^{(i)}}^2\right] = N_\ell \mathbb{C}\text{ov}[{Z_\ell}^2, {Z_{\ell-1}}^2] \\
&= N_\ell \mathbb{C}\text{ov}[{Q_\ell}^2 - 2Q_\ell\mu_{1,\ell} + {\mu_{1,\ell}}^2, {Q_{\ell-1}}^2 - 2Q_{\ell-1}\mu_{1,\ell-1} + {\mu_{1,\ell-1}}^2] \\
&= N_\ell\Bigg(\mathbb{C}\text{ov}[{Q_\ell}^2, {Q_{\ell-1}}^2] - 2\mu_{1,\ell-1}\mathbb{C}\text{ov}[{Q_\ell}^2, Q_{\ell-1}] - 2\mu_{1,\ell}\mathbb{C}\text{ov}[Q_\ell, {Q_{\ell-1}}^2] \\
&+ 4\mu_{1,\ell}\mu_{1,\ell-1}\mathbb{C}\text{ov}[Q_\ell, Q_{\ell-1}]\Bigg).
\end{aligned}$$

$$\begin{aligned}
&\text{(c2) } \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}, \left(\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}\right)^2\right] = \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}, \sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right] \\
&= \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}, \sum_{i=1}^{N_\ell} {Z_{\ell-1}^{(i)}}^2\right] + \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}, \sum_{i=1}^{N_\ell}\sum_{j=1, j\neq i}^{N_\ell} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right] \\
&= \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}, \sum_{i=1}^{N_\ell} {Z_{\ell-1}^{(i)}}^2\right] = (c1)
\end{aligned}$$

since (making use of centered Z and independence of $Z_{\ell-1}^{(i)}$ to $Z_{\ell-1}^{(j)}$)

$$\begin{aligned}
&\mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}, \sum_{i=1}^{N_\ell}\sum_{j=1, j\neq i}^{N_\ell} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right] \\
&= \mathbb{E}\left[\left(\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}\right)\sum_{i=1}^{N_\ell}\sum_{j=1, j\neq i}^{N_\ell} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right] - \mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}\right]\mathbb{E}\left[\sum_{i=1}^{N_\ell}\sum_{j=1, j\neq i}^{N_\ell} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right] \\
&= \mathbb{E}\left[\left(\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}\right)\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)} \sum_{j=1, j\neq i}^{N_\ell} Z_{\ell-1}^{(j)}\right] - \mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}\right]\mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)} \sum_{j=1, j\neq i}^{N_\ell} Z_{\ell-1}^{(j)}\right] \\
&= \mathbb{E}\left[\left(\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}\right)\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}\right]\mathbb{E}\left[\sum_{j=1, j\neq i}^{N_\ell} Z_{\ell-1}^{(j)}\right] - \mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}\right]\mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}\right] \\
&\mathbb{E}\left[\sum_{j=1, j\neq i}^{N_\ell} Z_{\ell-1}^{(j)}\right] = \mathbb{E}\left[\left(\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}\right)\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}\right]\sum_{j=1, j\neq i}^{N_\ell} \underbrace{\mathbb{E}[Z_{\ell-1}^{(j)}]}_{=0} \\
&\quad - \mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)^2}\right]\mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}\right]\sum_{j=1, j\neq i}^{N_\ell} \underbrace{\mathbb{E}[Z_{\ell-1}^{(j)}]}_{=0} = 0
\end{aligned}$$

$$\text{(c3) } \mathbb{C}\text{ov}\left[\left(\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\right)^2, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right] = \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(i)} Z_\ell^{(j)}, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right]$$

$$= \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)2} + \sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(i)} Z_\ell^{(j)}, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right]$$

$$= \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)2}, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right] + \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(i)} Z_\ell^{(j)}, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right]$$

$$= \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)2}, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right] + \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)} \sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(j)}, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right]$$

$$= \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)2}, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right] + \mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)} \sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(j)} \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right]$$

$$- \mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)} \sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(j)}\right] \mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right]$$

$$= \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)2}, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right] + \mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)} \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right] \mathbb{E}\left[\sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(j)}\right]$$

$$- \mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\right] \mathbb{E}\left[\sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(j)}\right] \mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right]$$

$$= \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)2}, \sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}{}^2\right] = (c1), \text{since } \mathbb{E}[Z_\ell^{(i)}] = 0.$$

(c4) $$\mathbb{C}\text{ov}\left[\left(\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\right)^2, \left(\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}\right)^2\right]$$

$$= \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(i)} Z_\ell^{(j)}, \sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right]$$

$$= \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} {Z_\ell^{(i)}}^2 + \sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(i)} Z_\ell^{(j)}, \sum_{i=1}^{N_\ell} {Z_{\ell-1}^{(i)}}^2 + \sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right]$$

$$= \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} {Z_\ell^{(i)}}^2, \sum_{i=1}^{N_\ell} {Z_{\ell-1}^{(i)}}^2\right] + \underbrace{\mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} {Z_\ell^{(i)}}^2, \sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right]}_{=0}$$

$$+ \underbrace{\mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(i)} Z_\ell^{(j)}, \sum_{i=1}^{N_\ell} {Z_{\ell-1}^{(i)}}^2\right]}_{=0}$$

$$+ \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(i)} Z_\ell^{(j)}, \sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right]$$

$$= \underbrace{\mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell} {Z_\ell^{(i)}}^2, \sum_{i=1}^{N_\ell} {Z_{\ell-1}^{(i)}}^2\right]}_{=(c1)}$$

$$+ \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(i)} Z_\ell^{(j)}, \sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right]$$

$$= (c1) + \mathbb{C}\text{ov}\left[\sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_\ell^{(i)} Z_\ell^{(j)}, \sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right]$$

$$= (c1) + \sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} \mathbb{C}\text{ov}\left[Z_\ell^{(i)} Z_\ell^{(j)}, Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}\right]$$

$$= (c1) + \sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} \mathbb{E}[Z_\ell^{(i)} Z_\ell^{(j)} Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}] - \mathbb{E}[Z_\ell^{(i)} Z_\ell^{(j)}]\mathbb{E}[Z_{\ell-1}^{(i)} Z_{\ell-1}^{(j)}]$$

$$= (c1) + \sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} \mathbb{E}[Z_\ell^{(i)} Z_{\ell-1}^{(i)}]\mathbb{E}[Z_\ell^{(j)} Z_{\ell-1}^{(j)}] - \mathbb{E}[Z_\ell^{(i)}]\mathbb{E}[Z_\ell^{(j)}]\mathbb{E}[Z_{\ell-1}^{(i)}]\mathbb{E}[Z_{\ell-1}^{(j)}]$$

$$= (c1) + \sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell} \mathbb{E}[Z_\ell^{(i)} Z_{\ell-1}^{(i)}]\mathbb{E}[Z_\ell^{(j)} Z_{\ell-1}^{(j)}]$$

$$\begin{aligned}
\text{(c4) cont.} \quad &= (c1) + \sum_{i=1}^{N_\ell} \sum_{j=1, j\neq i}^{N_\ell} \mathbb{E}[Z_\ell^{(i)} Z_{\ell-1}^{(i)}] \mathbb{E}[Z_\ell^{(j)} Z_{\ell-1}^{(j)}] \\
&= (c1) + \sum_{i=1}^{N_\ell} \sum_{j=1, j\neq i}^{N_\ell} \mathbb{E}\left[(Q_\ell^{(i)} - \mu_{1,\ell})(Q_{\ell-1}^{(i)} - \mu_{1,\ell-1})\right] \\
&\quad \mathbb{E}\left[(Q_\ell^{(j)} - \mu_{1,\ell})(Q_{\ell-1}^{(j)} - \mu_{1,\ell-1})\right] \\
&= (c1) + \sum_{i=1}^{N_\ell} \sum_{j=1, j\neq i}^{N_\ell} \Big(\mathbb{E}[Q_\ell^{(i)} Q_{\ell-1}^{(i)}] - \mathbb{E}[Q_\ell^{(i)} \mu_{1,\ell-1}] \\
&\quad - \mathbb{E}[\mu_{1,\ell} Q_{\ell-1}^{(i)}] + \mathbb{E}[\mu_{1,\ell}\mu_{1,\ell-1}] \Big) \\
&\Big(\mathbb{E}[Q_\ell^{(j)} Q_{\ell-1}^{(j)}] - \mathbb{E}[Q_\ell^{(j)} \mu_{1,\ell-1}] - \mathbb{E}[\mu_{1,\ell} Q_{\ell-1}^{(j)}] + \mathbb{E}[\mu_{1,\ell}\mu_{1,\ell-1}] \Big) \\
&= (c1) + \sum_{i=1}^{N_\ell} \sum_{j=1, j\neq i}^{N_\ell} \Big(\mathbb{E}[Q_\ell^{(i)} Q_{\ell-1}^{(i)}] - \mathbb{E}[Q_\ell^{(i)}] \mu_{1,\ell-1} \\
&\quad - \mu_{1,\ell} \mathbb{E}[Q_{\ell-1}^{(i)}] + \mu_{1,\ell}\mu_{1,\ell-1} \Big) \\
&\Big(\mathbb{E}[Q_\ell^{(j)} Q_{\ell-1}^{(j)}] - \mathbb{E}[Q_\ell^{(j)}] \mu_{1,\ell-1} - \mu_{1,\ell} \mathbb{E}[Q_{\ell-1}^{(j)}] + \mu_{1,\ell}\mu_{1,\ell-1} \Big) \\
&= (c1) + \sum_{i=1}^{N_\ell} \sum_{j=1, j\neq i}^{N_\ell} \Big(\mathbb{E}[Q_\ell^{(i)} Q_{\ell-1}^{(i)}] \\
&\quad - \mu_{1,\ell}\mu_{1,\ell-1} - \mu_{1,\ell}\mu_{1,\ell-1} + \mu_{1,\ell}\mu_{1,\ell-1} \Big) \\
&\Big(\mathbb{E}[Q_\ell^{(j)} Q_{\ell-1}^{(j)}] - \mu_{1,\ell}\mu_{1,\ell-1} - \mu_{1,\ell}\mu_{1,\ell-1} + \mu_{1,\ell}\mu_{1,\ell-1} \Big) \\
&= (c1) + \sum_{i=1}^{N_\ell} \sum_{j=1, j\neq i}^{N_\ell} \Big(\mathbb{E}[Q_\ell^{(i)} Q_{\ell-1}^{(i)}] - \mu_{1,\ell}\mu_{1,\ell-1} \Big) \\
&\quad \Big(\mathbb{E}[Q_\ell^{(j)} Q_{\ell-1}^{(j)}] - \mu_{1,\ell}\mu_{1,\ell-1} \Big) \\
&= (c1) + N_\ell N_{\ell-1} (\mathbb{E}[Q_\ell Q_{\ell-1}] - \mu_{1,\ell}\mu_{1,\ell-1})^2.
\end{aligned}$$

Combine all terms and simplify:

$$\begin{aligned}
\mathbb{C}\mathrm{ov}[\widehat{\mu}_{2,\ell},\widehat{\mu}_{2,\ell-1}] &= \frac{1}{(N_{\ell-1})^2}N_\ell\bigg(\mathbb{C}\mathrm{ov}[{Q_\ell}^2,{Q_{\ell-1}}^2]-2\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[{Q_\ell}^2,Q_{\ell-1}]\\
&\quad -2\mu_{1,\ell}\mathbb{C}\mathrm{ov}[Q_\ell,{Q_{\ell-1}}^2]+4\mu_{1,\ell}\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[Q_\ell,Q_{\ell-1}]\bigg) && \text{(c1)}\\
&\quad -\frac{1}{N_\ell(N_{\ell-1})^2}N_\ell\bigg(\mathbb{C}\mathrm{ov}[{Q_\ell}^2,{Q_{\ell-1}}^2]-2\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[{Q_\ell}^2,Q_{\ell-1}]\\
&\quad -2\mu_{1,\ell}\mathbb{C}\mathrm{ov}[Q_\ell,{Q_{\ell-1}}^2]+4\mu_{1,\ell}\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[Q_\ell,Q_{\ell-1}]\bigg) && \text{(c2)}\\
&\quad -\frac{1}{N_\ell(N_{\ell-1})^2}N_\ell\bigg(\mathbb{C}\mathrm{ov}[{Q_\ell}^2,{Q_{\ell-1}}^2]-2\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[{Q_\ell}^2,Q_{\ell-1}]\\
&\quad -2\mu_{1,\ell}\mathbb{C}\mathrm{ov}[Q_\ell,{Q_{\ell-1}}^2]+4\mu_{1,\ell}\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[Q_\ell,Q_{\ell-1}]\bigg) && \text{(c3)}\\
&\quad +\frac{1}{N_\ell^2(N_{\ell-1})^2}\bigg[N_\ell\bigg(\mathbb{C}\mathrm{ov}[{Q_\ell}^2,{Q_{\ell-1}}^2]-2\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[{Q_\ell}^2,Q_{\ell-1}]\\
&\quad -2\mu_{1,\ell}\mathbb{C}\mathrm{ov}[Q_\ell,{Q_{\ell-1}}^2]+4\mu_{1,\ell}\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[Q_\ell,Q_{\ell-1}]\bigg)\\
&\quad +N_\ell(N_{\ell-1})(\mathbb{E}[Q_\ell Q_{\ell-1}]-\mu_{1,\ell}\mu_{1,\ell-1})^2\bigg] && \text{(c4)}\\
&= \frac{N_\ell^2-2N_\ell+1}{N_\ell(N_{\ell-1})^2}\bigg(\mathbb{C}\mathrm{ov}[{Q_\ell}^2,{Q_{\ell-1}}^2]-2\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[{Q_\ell}^2,Q_{\ell-1}]\\
&\quad -2\mu_{1,\ell}\mathbb{C}\mathrm{ov}[Q_\ell,{Q_{\ell-1}}^2]+4\mu_{1,\ell}\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[Q_\ell,Q_{\ell-1}]\bigg)\\
&\quad +\frac{1}{N_\ell(N_{\ell-1})}(\mathbb{E}[Q_\ell Q_{\ell-1}]-\mu_{1,\ell}\mu_{1,\ell-1})^2\\
&= \frac{1}{N_\ell}\bigg(\mathbb{C}\mathrm{ov}[{Q_\ell}^2,{Q_{\ell-1}}^2]-2\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[{Q_\ell}^2,Q_{\ell-1}]\\
&\quad -2\mu_{1,\ell}\mathbb{C}\mathrm{ov}[Q_\ell,{Q_{\ell-1}}^2]+4\mu_{1,\ell}\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[Q_\ell,Q_{\ell-1}]\bigg)\\
&\quad +\frac{1}{N_\ell(N_{\ell-1})}(\mathbb{E}[Q_\ell Q_{\ell-1}]-\mu_{1,\ell}\mu_{1,\ell-1})^2.
\end{aligned}$$

Substitute covariance term and μ by expected value

$$\begin{aligned}
\mathbb{C}\mathrm{ov}[\widehat{\mu}_{2,\ell}, \widehat{\mu}_{2,\ell-1}] &= \frac{1}{N_\ell}\Big(\mathbb{C}\mathrm{ov}[{Q_\ell}^2, {Q_{\ell-1}}^2] - 2\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[{Q_\ell}^2, Q_{\ell-1}] \\
&- 2\mu_{1,\ell}\mathbb{C}\mathrm{ov}[Q_\ell, {Q_{\ell-1}}^2] + 4\mu_{1,\ell}\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[Q_\ell, Q_{\ell-1}]\Big) \\
&+ \frac{1}{N_\ell(N_{\ell-1})}(\mathbb{E}[Q_\ell Q_{\ell-1}] - \mu_{1,\ell}\mu_{1,\ell-1})^2 \\
&= \frac{1}{N_\ell}\Big(\mathbb{E}[{Q_\ell}^2{Q_{\ell-1}}^2] - \mathbb{E}[{Q_\ell}^2]\mathbb{E}[{Q_{\ell-1}}^2] \\
&- 2\mathbb{E}[Q_{\ell-1}](\mathbb{E}[{Q_\ell}^2 Q_{\ell-1}] - \mathbb{E}[{Q_\ell}^2]\mathbb{E}[Q_{\ell-1}]) \\
&- 2\mathbb{E}[Q_\ell](\mathbb{E}[Q_\ell {Q_{\ell-1}}^2] - \mathbb{E}[Q_\ell]\mathbb{E}[{Q_{\ell-1}}^2]) \\
&+ 4\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}](\mathbb{E}[Q_\ell Q_{\ell-1}] - \mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}])\Big) \\
&+ \frac{1}{N_\ell(N_{\ell-1})}\Big(\mathbb{E}[Q_\ell Q_{\ell-1}]^2 - 2\mathbb{E}[Q_\ell Q_{\ell-1}]\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}] \\
&+ \mathbb{E}[Q_\ell]^2\mathbb{E}[Q_{\ell-1}]^2\Big) \\
&= \frac{1}{N_\ell}\Big(\mathbb{E}[{Q_\ell}^2{Q_{\ell-1}}^2] - \mathbb{E}[{Q_\ell}^2]\mathbb{E}[{Q_{\ell-1}}^2] \\
&- 2\mathbb{E}[Q_{\ell-1}]\mathbb{E}[{Q_\ell}^2 Q_{\ell-1}] + 2\mathbb{E}[Q_{\ell-1}]^2\mathbb{E}[{Q_\ell}^2] \\
&- 2\mathbb{E}[Q_\ell]\mathbb{E}[Q_\ell {Q_{\ell-1}}^2] + 2\mathbb{E}[Q_\ell]^2\mathbb{E}[{Q_{\ell-1}}^2] \\
&+ 4\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}]\mathbb{E}[Q_\ell Q_{\ell-1}] - 4\mathbb{E}[Q_\ell]^2\mathbb{E}[Q_{\ell-1}]^2)\Big) \\
&+ \frac{1}{N_\ell(N_{\ell-1})}\Big(\mathbb{E}[Q_\ell Q_{\ell-1}]^2 - 2\mathbb{E}[Q_\ell Q_{\ell-1}]\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}] \\
&+ (\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}])^2\Big).
\end{aligned}$$

$\square$

C.6 Proof: Unbiased estimator for expected value of product of means

Referenced in Lemma 12.

Proof.

$$
\begin{aligned}
\mathbb{E}[\widehat{\mu}_{1,\ell}\widehat{\mu}_{1,\ell-1}] &= \mathbb{E}\left[\frac{N_\ell}{N_\ell-1}(\widehat{\mu}_{1,\ell}\widehat{\mu}_{1,\ell-1})_{\text{biased}} - \frac{1}{N_\ell-1}\widehat{\mu}_{1,\ell}[Q_\ell Q_{\ell-1}]\right] \\
&= \frac{N_\ell}{N_\ell-1}\mathbb{E}\left[(\widehat{\mu}_{1,\ell}\widehat{\mu}_{1,\ell-1})_{\text{biased}}\right] - \frac{1}{N_\ell-1}\mathbb{E}\left[\widehat{\mu}_{1,\ell}[Q_\ell Q_{\ell-1}]\right] \\
&= \frac{N_\ell}{N_\ell-1}\mathbb{E}\left[\frac{1}{N_\ell^2}\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Q_\ell^{(i)}Q_{\ell-1}^{(j)}\right] - \frac{1}{N_\ell-1}\mathbb{E}\left[\frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Q_\ell^{(i)}Q_{\ell-1}^{(i)}\right] \\
&= \frac{N_\ell}{N_\ell-1}\frac{1}{N_\ell^2}\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell}\mathbb{E}\left[Q_\ell^{(i)}Q_{\ell-1}^{(j)}\right] - \frac{1}{N_\ell-1}\frac{1}{N_\ell}\sum_{i=1}^{N_\ell}\mathbb{E}\left[Q_\ell^{(i)}Q_{\ell-1}^{(i)}\right] \\
&= \frac{N_\ell}{N_\ell-1}\frac{1}{N_\ell^2}\sum_{i=1}^{N_\ell}\mathbb{E}\left[Q_\ell^{(i)}Q_{\ell-1}^{(i)}\right] \qquad \text{(C.12)} \\
&\quad + \frac{N_\ell}{N_\ell-1}\frac{1}{N_\ell^2}\sum_{i=1}^{N_\ell}\sum_{j=1,j\neq i}^{N_\ell}\mathbb{E}\left[Q_\ell^{(i)}\right]\mathbb{E}\left[Q_{\ell-1}^{(j)}\right] \\
&- \frac{1}{N_\ell-1}\frac{1}{N_\ell}\sum_{i=1}^{N_\ell}\mathbb{E}\left[Q_\ell^{(i)}Q_{\ell-1}^{(i)}\right] \\
&= \frac{1}{N_\ell-1}\mathbb{E}\left[Q_\ell Q_{\ell-1}\right] + \mathbb{E}\left[Q_\ell\right]\mathbb{E}\left[Q_{\ell-1}\right] - \frac{1}{N_\ell-1}\mathbb{E}\left[Q_\ell Q_{\ell-1}\right] \\
&= \mathbb{E}\left[Q_\ell\right]\mathbb{E}\left[Q_{\ell-1}\right].
\end{aligned}
$$

□

C.7 Proof: Unbiased estimator for expected value of triple product of means

Referenced in Lemma 13.

Proof.

$$
\begin{aligned}
\mathbb{E}\Big[\widehat{\mu}_{1,\ell_1}\widehat{\mu}_{1,\ell_2}\widehat{\mu}_{1,\ell_3}\Big] &= \mathbb{E}\Big[\frac{N_\ell^2}{(N_\ell-1)(N_\ell-2)}(\widehat{\mu}_{1,\ell_1}\widehat{\mu}_{1,\ell_2}\widehat{\mu}_{1,\ell_3})_{\text{biased}} \\
&- \frac{1}{(N_\ell-1)(N_\ell-2)}\widehat{\mu}_{1,\ell}[Q_{\ell_1}Q_{\ell_2}Q_{\ell_3}] - \frac{1}{N_\ell-2}\Big(\widehat{\mu}_{1,\ell}[Q_{\ell_1}Q_{\ell_2}]\widehat{\mu}_{1,\ell}[Q_{\ell_3}] \\
&+ \widehat{\mu}_{1,\ell}[Q_{\ell_1}Q_{\ell_3}]\widehat{\mu}_{1,\ell}[Q_{\ell_2}] + \widehat{\mu}_{1,\ell}[Q_{\ell_2}Q_{\ell_3}]\widehat{\mu}_{1,\ell}[Q_{\ell_1}]\Big)\Big] \\
&= \frac{N_\ell^2}{(N_\ell-1)(N_\ell-2)}\mathbb{E}\Big[(\widehat{\mu}_{1,\ell_1}\widehat{\mu}_{1,\ell_2}\widehat{\mu}_{1,\ell_3})_{\text{biased}}\Big] \\
&- \frac{1}{N_\ell-2}\Big(\mathbb{E}\Big[\widehat{\mu}_{1,\ell}[Q_{\ell_1}Q_{\ell_2}]\widehat{\mu}_{1,\ell}[Q_{\ell_3}]\Big] \\
&- \frac{1}{(N_\ell-1)(N_\ell-2)}\mathbb{E}\Big[\widehat{\mu}_{1,\ell}[Q_{\ell_1}Q_{\ell_2}Q_{\ell_3}]\Big] + \mathbb{E}\Big[\widehat{\mu}_{1,\ell}[Q_{\ell_1}Q_{\ell_3}]\widehat{\mu}_{1,\ell}[Q_{\ell_2}]\Big] \\
&+ \mathbb{E}\Big[\widehat{\mu}_{1,\ell}[Q_{\ell_2}Q_{\ell_3}]\widehat{\mu}_{1,\ell}[Q_{\ell_1}]\Big]\Big) \\
&= \frac{N_\ell^2}{(N_\ell-1)(N_\ell-2)}\frac{1}{N_\ell^3}\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell}\sum_{k=1}^{N_\ell}\mathbb{E}[Q_{\ell_1}^{(i)}Q_{\ell_2}^{(j)}Q_{\ell_3}^{(k)}] \\
&- \frac{1}{(N_\ell-1)(N_\ell-2)}\mathbb{E}\Big[Q_{\ell_1}Q_{\ell_2}Q_{\ell_3}\Big] \\
&- \frac{1}{N_\ell-2}\Big(\mathbb{E}\Big[Q_{\ell_1}Q_{\ell_2}\Big]\mathbb{E}\Big[Q_{\ell_3}\Big] + \mathbb{E}\Big[Q_{\ell_1}Q_{\ell_3}\Big]\mathbb{E}\Big[Q_{\ell_2}\Big] + \mathbb{E}\Big[Q_{\ell_2}Q_{\ell_3}\Big]\mathbb{E}\Big[Q_{\ell_1}\Big]\Big) \\
&= \frac{1}{N_\ell(N_\ell-1)(N_\ell-2)}\Big(\\
&\quad N_\ell(N_\ell-1)(N_\ell-2)\mu_{1,\ell_1}\mu_{1,\ell_2}\mu_{1,\ell_3} \text{ (case: } i \neq j \neq k) \\
&+ N_\ell(N_\ell-1)\mathbb{E}[Q_{\ell_1}Q_{\ell_2}]\mathbb{E}[Q_{\ell_3}] \text{ (case: } i == j \neq k) \\
&+ N_\ell(N_\ell-1)\mathbb{E}[Q_{\ell_1}Q_{\ell_3}]\mathbb{E}[Q_{\ell_2}] \text{ (case: } i == k \neq j) \\
&+ N_\ell(N_\ell-1)\mathbb{E}[Q_{\ell_2}Q_{\ell_3}]\mathbb{E}[Q_{\ell_1}] \text{ (case: } j == k \neq i) \\
&+ N_\ell\mathbb{E}[Q_{\ell_1}Q_{\ell_2}Q_{\ell_3}]\Big) - \frac{1}{(N_\ell-1)(N_\ell-2)}\mathbb{E}\Big[Q_{\ell_1}Q_{\ell_2}Q_{\ell_3}\Big] \\
&- \frac{1}{N_\ell-2}\Big(\mathbb{E}\Big[Q_{\ell_1}Q_{\ell_2}\Big]\mathbb{E}\Big[Q_{\ell_3}\Big] + \mathbb{E}\Big[Q_{\ell_1}Q_{\ell_3}\Big]\mathbb{E}\Big[Q_{\ell_2}\Big] + \mathbb{E}\Big[Q_{\ell_2}Q_{\ell_3}\Big]\mathbb{E}\Big[Q_{\ell_1}\Big]\Big) \\
&= \mu_{1,\ell_1}\mu_{1,\ell_2}\mu_{1,\ell_3}.
\end{aligned}
\tag{C.13}
$$

□

C.8 Proof: Unbiased estimator for expected value of quadruple product of means

Referenced in Lemma 14.

Proof.

$$
\begin{aligned}
&\mathbb{E}\Big[\widehat{\mu}_{1,\ell}^2\widehat{\mu}_{1,\ell-1}^2\Big] = \mathbb{E}\Big[\frac{N_\ell^3}{(N_\ell-1)(N_\ell-2)(N_\ell-3)}(\widehat{\mu}_{1,\ell}^2\widehat{\mu}_{1,\ell-1}^2)_{\text{biased}} \\
&- \frac{1}{N_\ell-3}\Big(\widehat{\mu}_{1,\ell}[{Q_\ell}^2]\widehat{\mu}_{1,\ell}[Q_{\ell-1}]^2 + 4\widehat{\mu}_{1,\ell}[Q_\ell Q_{\ell-1}]\widehat{\mu}_{1,\ell}[Q_\ell]\widehat{\mu}_{1,\ell}[Q_{\ell-1}] \\
&+ \widehat{\mu}_{1,\ell}[Q_\ell]^2\widehat{\mu}_{1,\ell}[{Q_{\ell-1}}^2]\Big) \\
&- \frac{1}{(N_\ell-2)(N_\ell-3)}\Big(\widehat{\mu}_{1,\ell}[{Q_\ell}^2]\widehat{\mu}_{1,\ell}[{Q_{\ell-1}}^2] + 2\widehat{\mu}_{1,\ell}[Q_\ell Q_{\ell-1}]^2 \\
&+ 2\widehat{\mu}_{1,\ell}[{Q_\ell}^2 Q_{\ell-1}]\widehat{\mu}_{1,\ell}[Q_{\ell-1}] + 2\widehat{\mu}_{1,\ell}[Q_\ell]\widehat{\mu}_{1,\ell}[Q_\ell {Q_{\ell-1}}^2]\Big) \\
&- \frac{1}{(N_\ell-1)(N_\ell-2)(N_\ell-3)}\widehat{\mu}_{1,\ell}[{Q_\ell}^2{Q_{\ell-1}}^2]\Big] \\
&= \frac{N_\ell^3}{(N_\ell-1)(N_\ell-2)(N_\ell-3)}\frac{1}{N_\ell^4}\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell}\sum_{k=1}^{N_\ell}\sum_{h=1}^{N_\ell}\mathbb{E}[Q_\ell^{(i)}Q_\ell^{(j)}Q_{\ell-1}^{(k)}Q_{\ell-1}^{(h)}] \\
&- \frac{1}{N_\ell-3}\Big(\mathbb{E}[{Q_\ell}^2]\mathbb{E}[Q_{\ell-1}]^2 + 4\mathbb{E}[Q_\ell Q_{\ell-1}]\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}] + \mathbb{E}[Q_\ell]^2\mathbb{E}[{Q_{\ell-1}}^2]\Big) \\
&- \frac{1}{(N_\ell-2)(N_\ell-3)}\Big(\mathbb{E}[{Q_\ell}^2]\mathbb{E}[{Q_{\ell-1}}^2] + 2\mathbb{E}[Q_\ell Q_{\ell-1}]^2 \\
&+ 2\mathbb{E}[{Q_\ell}^2 Q_{\ell-1}]\mathbb{E}[Q_{\ell-1}] + 2\mathbb{E}[Q_\ell]\mathbb{E}[Q_\ell {Q_{\ell-1}}^2]\Big) \\
&- \frac{1}{(N_\ell-1)(N_\ell-2)(N_\ell-3)}\mathbb{E}[{Q_\ell}^2{Q_{\ell-1}}^2] \\
&= \frac{1}{N_\ell(N_\ell-1)(N_\ell-2)(N_\ell-3)}\Big(N_\ell(N_\ell-1)(N_\ell-2)(N_\ell-3){\mu_{1,\ell}}^2{\mu_{1,\ell-1}}^2 \\
&\text{(one pair: ij, ik, ih, jk, jh, kh :)} + N_\ell(N_\ell-1)(N_\ell-2) \\
&\Big(\mathbb{E}[{Q_\ell}^2]\mathbb{E}[Q_{\ell-1}]^2 + \mathbb{E}[Q_\ell Q_{\ell-1}]\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}] \\
&+ \mathbb{E}[Q_\ell Q_{\ell-1}]\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}] + \mathbb{E}[Q_\ell]\mathbb{E}[Q_\ell Q_{\ell-1}]\mathbb{E}[Q_{\ell-1}] \\
&+ \mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}]\mathbb{E}[Q_\ell Q_{\ell-1}] + \mathbb{E}[Q_\ell]^2\mathbb{E}[{Q_{\ell-1}}^2]\Big) \\
&\text{(two pairs: ih kh, ik jh, ih jk)} + N_\ell(N_\ell-1) \\
&\Big(\mathbb{E}[{Q_\ell}^2]\mathbb{E}[{Q_{\ell-1}}^2] + \mathbb{E}[Q_\ell Q_{\ell-1}]^2 + \mathbb{E}[Q_\ell Q_{\ell-1}]^2\Big)
\end{aligned}
\tag{C.14}
$$

$$
\begin{aligned}
&\text{(triplets: ijk, ijh, jhk, ihk)} + N_\ell(N_\ell-1)\\
&\Big(\mathbb{E}[{Q_\ell}^2 Q_{\ell-1}]\mathbb{E}[Q_{\ell-1}] + \mathbb{E}[{Q_\ell}^2 Q_{\ell-1}]\mathbb{E}[Q_{\ell-1}]\\
&+ \mathbb{E}[Q_\ell]\mathbb{E}[Q_\ell {Q_{\ell-1}}^2] + \mathbb{E}[Q_\ell]\mathbb{E}[Q_\ell {Q_{\ell-1}}^2]\Big) \underbrace{+N_\ell \mathbb{E}[{Q_\ell}^2 {Q_{\ell-1}}^2]}_{\text{quadruple}}\Big)\\
&- \frac{1}{N_\ell-3}\Big(\mathbb{E}[{Q_\ell}^2]\mathbb{E}[Q_{\ell-1}]^2 + 4\mathbb{E}[Q_\ell Q_{\ell-1}]\mathbb{E}[Q_\ell]\mathbb{E}[Q_{\ell-1}] + \mathbb{E}[Q_\ell]^2\mathbb{E}[{Q_{\ell-1}}^2]\Big)\\
&- \frac{1}{(N_\ell-2)(N_\ell-3)}\Big(\mathbb{E}[{Q_\ell}^2]\mathbb{E}[{Q_{\ell-1}}^2] + 2\mathbb{E}[Q_\ell Q_{\ell-1}]^2 + 2\mathbb{E}[{Q_\ell}^2 Q_{\ell-1}]\mathbb{E}[Q_{\ell-1}]\\
&\qquad + 2\mathbb{E}[Q_\ell]\mathbb{E}[Q_\ell {Q_{\ell-1}}^2]\Big)\\
&- \frac{1}{(N_\ell-1)(N_\ell-2)(N_\ell-3)}\mathbb{E}[{Q_\ell}^2 {Q_{\ell-1}}^2] = {\mu_{1,\ell}}^2 {\mu_{1,\ell-1}}^2.
\end{aligned}
$$

□

C.9 Proof: Unbiased estimator for covariance of variance

Referenced in Lemma 15.

Proof. We use the previously proven unbiased estimators for products of expected values to show that the estimator is unbiased:

$$
\begin{aligned}
&\mathbb{E}\Big[\widehat{\mathbb{C}\mathrm{ov}}[\widehat{\mu}_{2,\ell}, \widehat{\mu}_{2,\ell-1}]\Big] = \frac{1}{N_\ell}\mathbb{E}\Big[\widehat{\mu}_1[\widehat{\mu}_{2,\ell}\widehat{\mu}_{2,\ell-1}]\Big]\\
&+ \frac{1}{N_\ell(N_\ell-1)}\Big(\mathbb{E}\Big[\widehat{\mu}_{1,\ell}[Q_\ell Q_{\ell-1}]\Big] - 2\mathbb{E}\Big[\widehat{\mu}_{1,\ell}[Q_\ell Q_{\ell-1}]\widehat{\mu}_{1,\ell}\widehat{\mu}_{1,\ell-1}\Big]\\
&- \mathbb{E}\Big[(\widehat{\mu}_{1,\ell}\widehat{\mu}_{1,\ell-1})^2\Big]\Big) \qquad \text{(C.15)}\\
&= \frac{1}{N_\ell}\mathbb{E}[\widehat{\mu}_{2,\ell}\widehat{\mu}_{2,\ell-1}]\\
&+ \frac{1}{N_\ell(N_\ell-1)}\Big(\mathbb{E}[Q_\ell Q_{\ell-1}] - 2\mathbb{E}[Q_\ell Q_{\ell-1}]\mu_{1,\ell}\mu_{1,\ell-1} - (\mu_{1,\ell}\mu_{1,\ell-1})^2\Big).
\end{aligned}
$$

□

C.10 Proof: Unbiased estimator for multilevel variance of variance

Referenced in Lemma 16.

Proof. Using Lemma 8 and Lemma 15, we have

$$\begin{aligned}\mathbb{E}\left[\widehat{\mu}_2[\widehat{\mu}_{2,\mathrm{ML}}]\right] &= \sum_{\ell=1}^{L} \mathbb{E}\left[\widehat{\mu}_2[\widehat{\mu}_{2,\ell}]\right] + \mathbb{E}\left[\widehat{\mu}_2[\widehat{\mu}_{2,\ell-1}]\right] - 2\mathbb{E}\left[\widehat{\mathbb{C}\mathrm{ov}}[\widehat{\mu}_{2,\ell},\widehat{\mu}_{2,\ell-1}]\right] \\ &= \sum_{\ell=1}^{L} \mathbb{V}[\widehat{\mu}_{2,\ell}] + \mathbb{V}[\widehat{\mu}_{2,\ell-1}] - 2\mathbb{C}\mathrm{ov}[\widehat{\mu}_{2,\ell},\widehat{\mu}_{2,\ell-1}].\end{aligned} \tag{C.16}$$

□

C.11 Proof: Same level $\mathbb{C}$ov$[\widehat{\mu}_{1,\ell},\widehat{\mu}_{2,\ell}]$

Referenced in Lemma 17.

Proof. To proof this relation we first need a few ingredients by following the proof given in [336]. Similar to the proof for the covariance term of Eq. (12.12) we use centered variables $Z_\ell^{(i)} = Q_\ell^{(i)} - \mu_{1,\ell}$ and $Z_{\ell-1}^{(i)} = Q_{\ell-1}^{(i)} - \mu_{1,\ell-1}$. Using that variable we know that

$$\begin{aligned}\widehat{\mu}_{2,\ell}[Z_\ell] &= \frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell}\left(Z_\ell^{(i)} - \frac{1}{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(j)}\right)^2 \\ &= \frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell}\left(Q_\ell^{(i)} - \mu_{1,\ell} - \frac{1}{N_\ell}\sum_{j=1}^{N_\ell} Q_\ell^{(j)} - \mu_{1,\ell}\right)^2 \\ &= \frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell}\left(Q_\ell^{(i)} - \frac{1}{N_\ell}\sum_{j=1}^{N_\ell} Q_\ell^{(j)}\right)^2 \\ &= \widehat{\mu}_{2,\ell}[Q_\ell].\end{aligned} \tag{C.17}$$

We furthermore know that

$$\begin{aligned}\widehat{\mu}_{2,\ell}[Z_\ell] &= \frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell}\left(Z_\ell^{(i)} - \frac{1}{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(j)}\right)^2 \\ &= \frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell}(Z_\ell^{(i)})^2 - \frac{N_\ell}{N_\ell - 1}\left(\frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\right)^2.\end{aligned} \tag{C.18}$$

Additionally, we will later on need the following two equalities for the product of centered random variables:

$$\begin{aligned}\mathbb{E}[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(i)}(Z_\ell^{(j)})^2] &= \mathbb{E}[\underbrace{N_\ell(Z_\ell^{(i)})^3}_{i=j} + \underbrace{N_\ell(N_\ell-1)Z_\ell^{(i)}(Z_\ell^{(j)})^2}_{i\neq j}] \\ &= N_\ell\mathbb{E}[(Z_\ell^{(i)})^3] + N_\ell(N_\ell-1)\underbrace{\mathbb{E}[Z_\ell^{(i)}]}_{=0}\mathbb{E}[(Z_\ell^{(j)})^2] \\ &= N_\ell\mu_{3,\ell}\end{aligned} \tag{C.19}$$

and

$$\begin{aligned}&\mathbb{E}[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell}\sum_{k=1}^{N_\ell} Z_\ell^{(i)}Z_\ell^{(j)}Z_\ell^{(k)}] \\ &= \mathbb{E}[\underbrace{N_\ell(Z_\ell^{(i)})^3}_{i=j=k} + 3\underbrace{N_\ell(N_\ell-1)(Z_\ell^{(i)})^2 Z_\ell^{(j)}}_{i\neq j=k\vee i=j\neq k\vee i\neq k=j} + N_\ell(N_\ell-1)(N_\ell-2)Z_\ell^{(i)}Z_\ell^{(j)}Z_\ell^{(k)}] \\ &= N_\ell\mathbb{E}[(Z_\ell^{(i)})^3] + 3N_\ell(N_\ell-1)\mathbb{E}[(Z_\ell^{(i)})^2]\underbrace{\mathbb{E}[Z_\ell^{(j)}]}_{=0} \\ &+ N_\ell(N_\ell-1)(N_\ell-2)\underbrace{\mathbb{E}[Z_\ell^{(i)}]}_{=0}\underbrace{\mathbb{E}[Z_\ell^{(j)}]}_{=0}\underbrace{\mathbb{E}[Z_\ell^{(k)}]}_{=0} \\ &= N_\ell\mu_{3,\ell}.\end{aligned} \tag{C.20}$$

Next, we use the following relationship

$$\begin{aligned}\mathbb{C}\mathrm{ov}[\widehat{\mu}_{1,\ell},\widehat{\mu}_{2,\ell}] &= \mathbb{E}[\widehat{\mu}_{1,\ell}\widehat{\mu}_{2,\ell}] - \mathbb{E}[\widehat{\mu}_{1,\ell}]\mathbb{E}[\widehat{\mu}_{2,\ell}] \\ &= \mathbb{E}[(\widehat{\mu}_{1,\ell}+\mu_{1,\ell}-\mu_{1,\ell})\widehat{\mu}_{2,\ell}] - \mu_{1,\ell}\mu_{2,\ell} \\ &= \mathbb{E}[(\widehat{\mu}_{1,\ell}-\mu_{1,\ell})\widehat{\mu}_{2,\ell} + \mu_{1,\ell}\widehat{\mu}_{2,\ell}] - \mu_{1,\ell}\mu_{2,\ell} \\ &= \mathbb{E}[(\widehat{\mu}_{1,\ell}-\mu_{1,\ell})\widehat{\mu}_{2,\ell}] + \mu_{1,\ell}\mathbb{E}[\widehat{\mu}_{2,\ell}] - \mu_{1,\ell}\mu_{2,\ell} \\ &= \mathbb{E}[(\widehat{\mu}_{1,\ell}-\mu_{1,\ell})\widehat{\mu}_{2,\ell}].\end{aligned} \tag{C.21}$$

Given Eq. (C.17) and Eq. (C.18), we can rewrite Eq. (C.21) in centered form

$$\begin{aligned}
&\mathbb{E}\left[(\widehat{\mu}_{1,\ell}[Q_\ell] - \mu_{1,\ell}[Q_\ell])\widehat{\mu}_{2,\ell}[Q_\ell]\right] = \mathbb{E}\left[\widehat{\mu}_{1,\ell}[Z_\ell^{(i)}]\widehat{\mu}_{2,\ell}[Z_\ell^{(i)}]\right] \\
&= \mathbb{E}\left[\left(\frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\right)\left(\frac{1}{N_\ell-1}\sum_{i=1}^{N_\ell}(Z_\ell^{(i)})^2 - \frac{N_\ell}{N_\ell-1}(\frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Z_\ell^{(i)})^2\right)\right] \\
&= \frac{1}{N_\ell}\frac{1}{N_\ell-1}\mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\left(\sum_{i=1}^{N_\ell}(Z_\ell^{(i)})^2 - \frac{1}{N_\ell}(\sum_{i=1}^{N_\ell} Z_\ell^{(i)})^2\right)\right] \\
&= \frac{1}{N_\ell}\frac{1}{N_\ell-1}\mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\sum_{i=1}^{N_\ell}(Z_\ell^{(i)})^2 - \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Z_\ell^{(i)}(\sum_{i=1}^{N_\ell} Z_\ell^{(i)})^2\right] \\
&= \frac{1}{N_\ell}\frac{1}{N_\ell-1}\mathbb{E}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(i)}(Z_\ell^{(j)})^2 - \frac{1}{N_\ell}\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell}\sum_{k=1}^{N_\ell} Z_\ell^{(i)}Z_\ell^{(j)}Z_\ell^{(k)}\right] \\
&= \frac{1}{N_\ell}\frac{1}{N_\ell-1}\mathbb{E}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(i)}(Z_\ell^{(j)})^2\right] \\
&- \frac{1}{N_\ell}\frac{1}{N_\ell}\frac{1}{N_\ell-1}\mathbb{E}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell}\sum_{k=1}^{N_\ell} Z_\ell^{(i)}Z_\ell^{(j)}Z_\ell^{(k)}\right].
\end{aligned} \tag{C.22}$$

Next, we can use the relations from Eq. (C.19) and Eq. (C.20) to finalize the proof

$$\begin{aligned}
\mathbb{C}\text{ov}[\widehat{\mu}_{1,\ell}, \widehat{\mu}_{2,\ell}] &= \frac{1}{N_\ell}\frac{1}{N_\ell-1}\mathbb{E}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(i)}(Z_\ell^{(j)})^2\right] \\
&- \frac{1}{N_\ell}\frac{1}{N_\ell}\frac{1}{N_\ell-1}\mathbb{E}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell}\sum_{k=1}^{N_\ell} Z_\ell^{(i)}Z_\ell^{(j)}Z_\ell^{(k)}\right] \\
&= \frac{1}{N_\ell}\frac{1}{N_\ell-1}N_\ell\mu_{3,\ell} - \frac{1}{N_\ell}\frac{1}{N_\ell}\frac{1}{N_\ell-1}N_\ell\mu_{3,\ell} \\
&= \frac{1}{N_\ell-1}\mu_{3,\ell} - \frac{1}{N_\ell(N_\ell-1)}\mu_{3,\ell} \\
&= \frac{\mu_{3,\ell}}{N_\ell}.
\end{aligned} \tag{C.23}$$

□

C.12 Proof: Lower level variance $\mathbb{C}\mathrm{ov}[\widehat{\mu}_{1,\ell}, \widehat{\mu}_{2,\ell-1}]$

Referenced in Lemma 18.

Proof. We need the following two relations for product of centered random variables for this proof:

$$\begin{aligned}
&\mathbb{E}[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(i)}(Z_{\ell-1}^{(j)})^2] = \mathbb{E}[\underbrace{N_\ell Z_\ell^{(i)}(Z_{\ell-1}^{(i)})^2}_{i=j} + \underbrace{N_\ell(N_\ell-1)Z_\ell^{(i)}(Z_{\ell-1}^{(j)})^2}_{i\neq j}] \\
&= N_\ell \mathbb{E}[Z_\ell^{(i)}(Z_{\ell-1}^{(i)})^2] + N_\ell(N_\ell-1)\underbrace{\mathbb{E}[Z_\ell^{(i)}]}_{=0}\mathbb{E}[(Z_{\ell-1}^{(j)})^2] \\
&= N_\ell \mathbb{E}[Z_\ell^{(i)}(Z_{\ell-1}^{(i)})^2] \\
&= N_\ell(\mathbb{C}\mathrm{ov}[Z_\ell^{(i)}, (Z_{\ell-1}^{(i)})^2] + \underbrace{\mathbb{E}[Z_\ell^{(i)}]}_{=0}\mathbb{E}[(Z_{\ell-1}^{(i)})^2]) \\
&= N_\ell(\mathbb{C}\mathrm{ov}[Q_\ell^{(i)} - \mu_{1,\ell}, (Q_{\ell-1}^{(i)} - \mu_{1,\ell-1})^2]) \\
&= N_\ell(\mathbb{C}\mathrm{ov}[Q_\ell^{(i)}, (Q_{\ell-1}^{(i)})^2 - 2Q_{\ell-1}^{(i)}\mu_{1,\ell-1} + (\mu_{1,\ell-1})^2]) \\
&= N_\ell(\mathbb{C}\mathrm{ov}[Q_\ell^{(i)}, (Q_{\ell-1}^{(i)})^2] - 2\mu_{1,\ell-1}\mathbb{C}\mathrm{ov}[Q_\ell^{(i)}, Q_{\ell-1}^{(i)}] + \underbrace{\mathbb{C}\mathrm{ov}[Q_\ell^{(i)}, (\mu_{1,\ell-1})^2])}_{=0} \qquad \text{(C.24)} \\
&= N_\ell\Big[\Big(\mathbb{E}[Q_\ell^{(i)}(Q_{\ell-1}^{(i)})^2] - \mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[(Q_{\ell-1}^{(i)})^2]\Big) \\
&- 2\mu_{1,\ell-1}\Big(\mathbb{E}[Q_\ell^{(i)}Q_{\ell-1}^{(i)}] - \mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[Q_{\ell-1}^{(i)}]\Big)\Big] \\
&= N_\ell\Big[\Big(\mathbb{E}[Q_\ell^{(i)}(Q_{\ell-1}^{(i)})^2] - \mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[(Q_{\ell-1}^{(i)})^2]\Big) \\
&- 2\mathbb{E}[Q_{\ell-1}^{(i)}]\Big(\mathbb{E}[Q_\ell^{(i)}Q_{\ell-1}^{(i)}] - \mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[Q_{\ell-1}^{(i)}]\Big)\Big] \\
&= N_\ell\Big[\mathbb{E}[Q_\ell^{(i)}(Q_{\ell-1}^{(i)})^2] - \mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[(Q_{\ell-1}^{(i)})^2] \\
&+ 2\mathbb{E}[Q_{\ell-1}^{(i)}]\mathbb{E}[Q_\ell^{(i)}Q_{\ell-1}^{(i)}] - 2\mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[Q_{\ell-1}^{(i)}]^2\Big]
\end{aligned}$$

and

$$\begin{aligned}
&\mathbb{E}[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell}\sum_{k=1}^{N_\ell} Z_\ell^{(i)} Z_{\ell-1}^{(j)} Z_{\ell-1}^{(k)}] = \mathbb{E}[\underbrace{N_\ell Z_\ell^{(i)}(Z_{\ell-1}^{(i)})^2}_{i=j=k} \\
&+ \underbrace{N_\ell(N_\ell-1)Z_\ell^{(i)}(Z_{\ell-1}^{(j)})^2}_{i\neq j=k} + \underbrace{N_\ell(N_\ell-1)Z_\ell^{(i)}Z_{\ell-1}^{(i)}Z_{\ell-1}^{(k)}}_{i=j\neq k} + \underbrace{N_\ell(N_\ell-1)Z_\ell^{(i)}(Z_{\ell-1}^{(j)})^2}_{i\neq k=j} \\
&+ N_\ell(N_\ell-1)(N_\ell-2)Z_\ell^{(i)}Z_{\ell-1}^{(j)}Z_{\ell-1}^{(k)}] \\
&= \mathbb{E}[N_\ell Z_\ell^{(i)}(Z_{\ell-1}^{(i)})^2] \\
&+ N_\ell(N_\ell-1)\underbrace{\mathbb{E}[Z_\ell^{(i)}]}_{=0}\mathbb{E}[(Z_{\ell-1}^{(j)})^2] + \mathbb{E}[Z_\ell^{(i)}Z_{\ell-1}^{(i)}]\underbrace{\mathbb{E}[Z_{\ell-1}^{(k)}]}_{=0} + \underbrace{\mathbb{E}[Z_\ell^{(i)}]}_{=0}\mathbb{E}[(Z_{\ell-1}^{(j)})^2] \\
&+ N_\ell(N_\ell-1)(N_\ell-2)\underbrace{\mathbb{E}[Z_\ell^{(i)}]}_{=0}\underbrace{\mathbb{E}[Z_{\ell-1}^{(j)}]}_{=0}\underbrace{\mathbb{E}[Z_{\ell-1}^{(k)}]}_{=0} \\
&= \mathbb{E}[N_\ell Z_\ell^{(i)}(Z_{\ell-1}^{(i)})^2] \\
&\overbrace{=}^{\text{(C.24)}} N_\ell\Big[\mathbb{E}[Q_\ell^{(i)}(Q_{\ell-1}^{(i)})^2] - \mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[(Q_{\ell-1}^{(i)})^2] \\
&- 2\mathbb{E}[Q_{\ell-1}^{(i)}]\mathbb{E}[Q_\ell^{(i)}Q_{\ell-1}^{(i)}] + 2\mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[Q_{\ell-1}^{(i)}]^2\Big].
\end{aligned} \tag{C.25}$$

Similarly to the first proof, we use the relation:

$$\begin{aligned}
\mathbb{C}\text{ov}[\widehat{\mu}_{1,\ell}, \widehat{\mu}_{2,\ell-1}] &= \mathbb{E}[\widehat{\mu}_{1,\ell}\widehat{\mu}_{2,\ell-1}] - \mathbb{E}[\widehat{\mu}_{1,\ell}]\mathbb{E}[\widehat{\mu}_{2,\ell-1}] \\
&= \mathbb{E}[(\widehat{\mu}_{1,\ell} + \mu_{1,\ell} - \mu_{1,\ell})\widehat{\mu}_{2,\ell-1}] - \mu_{1,\ell}\mu_{2,\ell-1} \\
&= \mathbb{E}[(\widehat{\mu}_{1,\ell} - \mu_{1,\ell})\widehat{\mu}_{2,\ell-1} + \mu_{1,\ell}\widehat{\mu}_{2,\ell-1}] - \mu_{1,\ell}\mu_{2,\ell-1} \\
&= \mathbb{E}[(\widehat{\mu}_{1,\ell} - \mu_{1,\ell})\widehat{\mu}_{2,\ell-1}] + \mu_{1,\ell}\mathbb{E}[\widehat{\mu}_{2,\ell-1}] - \mu_{1,\ell}\mu_{2,\ell-1} \\
&= \mathbb{E}[(\widehat{\mu}_{1,\ell} - \mu_{1,\ell})\widehat{\mu}_{2,\ell-1}].
\end{aligned} \tag{C.26}$$

Given Eq. (C.17) and Eq. (C.18), we can rewrite Eq. (C.26) in centered form:

$$
\begin{aligned}
&\mathbb{E}\left[(\widehat{\mu}_{1,\ell}[Q_\ell] - \mu_{1,\ell}[Q_\ell])\widehat{\mu}_{2,\ell-1}[Q_\ell]\right] = \mathbb{E}\left[\widehat{\mu}_{1,\ell}[Z_\ell^{(i)}]\widehat{\mu}_{2,\ell-1}[Z_\ell^{(i)}]\right] \\
&= \mathbb{E}\left[\left(\frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\right)\left(\frac{1}{N_\ell - 1}\sum_{i=1}^{N_\ell}(Z_{\ell-1}^{(i)})^2 - \frac{N_\ell}{N_\ell - 1}(\frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)})^2\right)\right] \\
&= \frac{1}{N_\ell}\frac{1}{N_\ell - 1}\mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\left(\sum_{i=1}^{N_\ell}(Z_{\ell-1}^{(i)})^2 - \frac{1}{N_\ell}(\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)})^2\right)\right] \\
&= \frac{1}{N_\ell}\frac{1}{N_\ell - 1}\mathbb{E}\left[\sum_{i=1}^{N_\ell} Z_\ell^{(i)}\sum_{i=1}^{N_\ell}(Z_{\ell-1}^{(i)})^2 - \frac{1}{N_\ell}\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)}(\sum_{i=1}^{N_\ell} Z_{\ell-1}^{(i)})^2\right] \\
&= \frac{1}{N_\ell}\frac{1}{N_\ell - 1}\mathbb{E}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(i)}(Z_{\ell-1}^{(j)})^2 - \frac{1}{N_\ell}\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell}\sum_{k=1}^{N_\ell} Z_\ell^{(i)}Z_{\ell-1}^{(j)}Z_{\ell-1}^{(k)}\right] \\
&= \frac{1}{N_\ell}\frac{1}{N_\ell - 1}\left(\mathbb{E}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(i)}(Z_{\ell-1}^{(j)})^2\right] - \frac{1}{N_\ell}\mathbb{E}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell}\sum_{k=1}^{N_\ell} Z_\ell^{(i)}Z_{\ell-1}^{(j)}Z_{\ell-1}^{(k)}\right]\right).
\end{aligned}
\tag{C.27}
$$

Again, we use the relation for the product of centered variables from Eq. (C.24) and Eq. (C.25):

$$
\begin{aligned}
\mathbb{C}\mathrm{ov}[\widehat{\mu}_{1,\ell}, \widehat{\mu}_{2,\ell-1}] &= \frac{1}{N_\ell}\frac{1}{N_\ell - 1}\mathbb{E}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell} Z_\ell^{(i)}(Z_{\ell-1}^{(j)})^2\right] \\
&- \frac{1}{N_\ell}\frac{1}{N_\ell}\frac{1}{N_\ell - 1}\mathbb{E}\left[\sum_{i=1}^{N_\ell}\sum_{j=1}^{N_\ell}\sum_{k=1}^{N_\ell} Z_\ell^{(i)}Z_{\ell-1}^{(j)}Z_{\ell-1}^{(k)}\right] \\
&= \frac{1}{N_\ell}\frac{1}{N_\ell - 1}N_\ell\Big[\mathbb{E}[Q_\ell^{(i)}(Q_{\ell-1}^{(i)})^2] - \mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[(Q_{\ell-1}^{(i)})^2] \\
&- 2\mathbb{E}[Q_{\ell-1}^{(i)}]\mathbb{E}[Q_\ell^{(i)}Q_{\ell-1}^{(i)}] - 2\mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[Q_{\ell-1}^{(i)}]^2\Big] \\
&- \frac{1}{N_\ell}\frac{1}{N_\ell}\frac{1}{N_\ell - 1}N_\ell\Big[\mathbb{E}[Q_\ell^{(i)}(Q_{\ell-1}^{(i)})^2] - \mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[(Q_{\ell-1}^{(i)})^2] \\
&- 2\mathbb{E}[Q_{\ell-1}^{(i)}]\mathbb{E}[Q_\ell^{(i)}Q_{\ell-1}^{(i)}] - 2\mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[Q_{\ell-1}^{(i)}]^2\Big] \\
&= \frac{1}{N_\ell}\Big[\mathbb{E}[Q_\ell^{(i)}(Q_{\ell-1}^{(i)})^2] - \mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[(Q_{\ell-1}^{(i)})^2] \\
&- 2\mathbb{E}[Q_{\ell-1}^{(i)}]\mathbb{E}[Q_\ell^{(i)}Q_{\ell-1}^{(i)}] + 2\mathbb{E}[Q_\ell^{(i)}]\mathbb{E}[Q_{\ell-1}^{(i)}]^2\Big].
\end{aligned}
\tag{C.28}
$$

□

C.13 Resource allocation problem of variance for two-level example

Referenced in Section 12.6.

We give an example of the resource allocation problem targeting the variance, Eq. (12.10), in Figs. C.1 and C.2, as a surface and contour plot, respectively. We compare OPT++ and NPSOL. Red circles show the optimization path. Red crosses show initial design, red diamonds show final design. The initial point is set as [5, 5] to start in a region of high gradients. The number in title corresponds to different seeds in Dakota. The equality constrained for $\epsilon^2_{\mathbb{V}}$ is visualized in magenta.

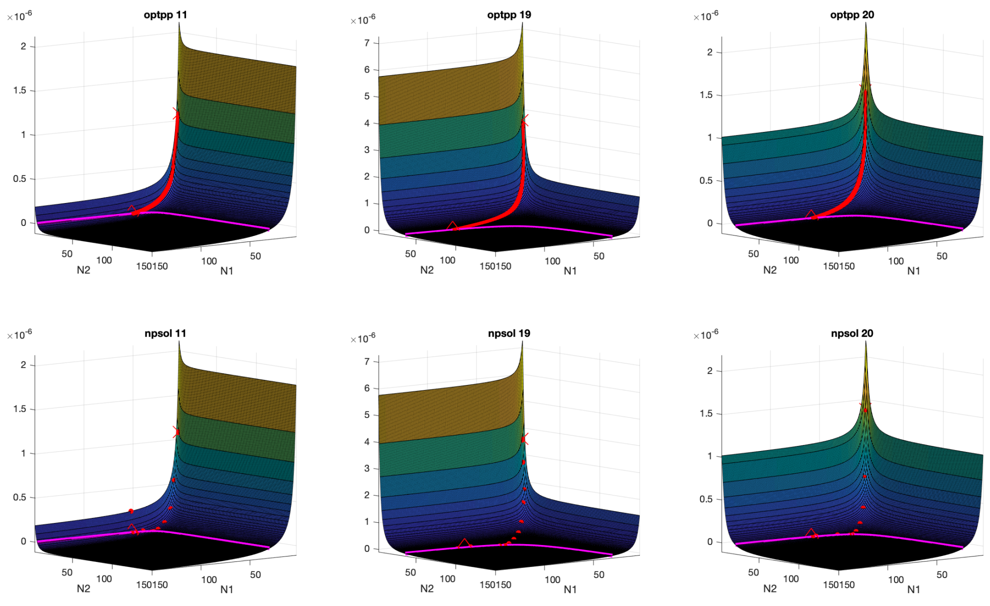


Figure C.1: Surface plot of the resource allocation objective function and variance constraint.

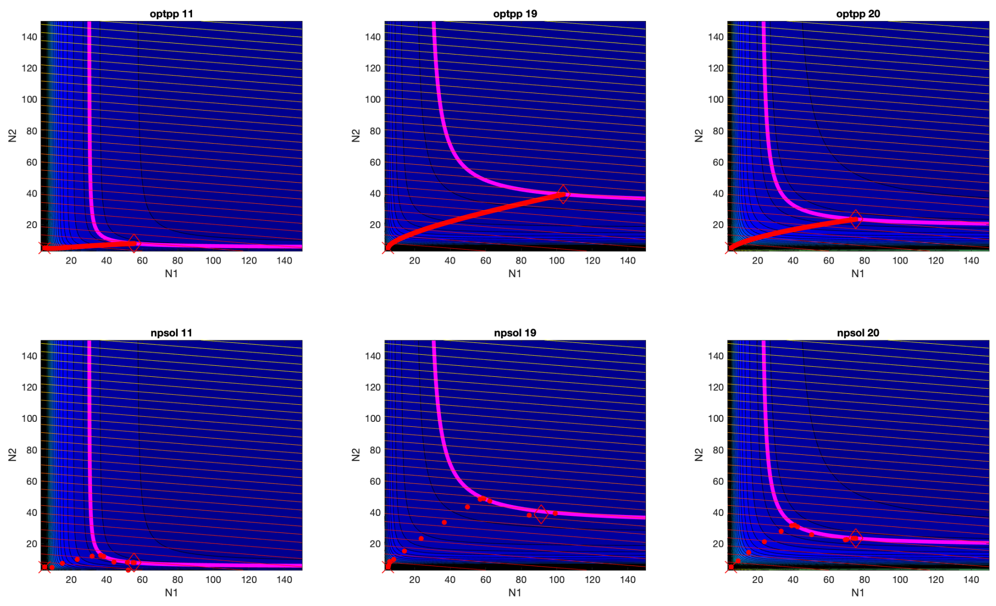


Figure C.2: Contour plot of the resource allocation objective function and variance constraint.

C.14 Problem 18 sampling: input file example

Referenced in Section 13.2.

```
environment,
    tabular_data
        tabular_data_file = 'dakota_problem18_scalarization.dat'
    method_pointer = 'OPTIM'

 method,
    id_method = 'OPTIM'
    model_pointer = 'OPTIM_M'
    centered_parameter_study
        steps_per_variable = 500
        step_vector = 0.0

 model,
    id_model = 'OPTIM_M'
    nested
      variables_pointer   = 'OPTIM_V'
      sub_method_pointer = 'UQ'
      responses_pointer   = 'OPTIM_R'
      primary_response_mapping    = 1. 3. 0. 0.
      secondary_response_mapping = 0. 0. 1.0 0.0

 variables,
    id_variables = 'OPTIM_V'
    continuous_design = 1
      initial_point      =  1
      descriptors        = 'x'

 responses,
    id_responses = 'OPTIM_R'
    objective_functions = 1
    nonlinear_inequality_constraints = 1
    nonlinear_inequality_lower_bounds = -1.e+50
    nonlinear_inequality_upper_bounds = 0
    no_gradients
    no_hessians

 ##########################
 # begin UQ specification #
 ##########################
 method,
    id_method = 'UQ'
    model_pointer = 'HIERARCH'
       multilevel_sampling
      pilot_samples = 10000 1000 100 50
      sample_type random
      final_moments standard
      max_iterations = 20
      convergence_tolerance 1.617509453025866e-05
        convergence_tolerance_type absolute
       convergence_tolerance_target variance_constraint
     allocation_target scalarization
       optimization
     qoi_aggregation max
     output silent

model,
   id_model = 'HIERARCH'
   variables_pointer = 'UQ_V'
   responses_pointer = 'UQ_R'
   surrogate hierarchical
     ordered_model_fidelities = 'MLModel'

model,
   id_model = 'MLModel'
   variables_pointer = 'UQ_V'
   interface_pointer = 'UQ_I'
   responses_pointer = 'UQ_R'
   simulation
       solution_level_control = 'Af'
               solution_level_cost = 0.01 1.0 0.1 0.001

variables,
   id_variables = 'UQ_V'
   continuous_design = 1
   uniform_uncertain = 1
     lower_bounds      =  -0.5
     upper_bounds      =  0.5
          descriptors       =  'xi'
   discrete_state_set real = 2
            num_set_values = 4 1
       initial_state = 1.5 0.0
       set_values = -6 1.0 1.1 1.5
       descriptors = 'Af' 'Ac'

interface,
   id_interface = 'UQ_I'
   direct
       analysis_driver = 'problem18'
        deactivate
                restart_file
                evaluation_cache

responses,
   id_responses = 'UQ_R'
   response_functions = 2
   no_gradients
   no_hessians
```

Listing C.1: Dakota input file for MLMC sampling of Problem 18 targeting the scalarization case of Section 13.3.4.

D The derivative-free stochastic nonlinear constrained optimization method SNOWPAC

D.1 Quantile sampling estimator

Referenced in Section 9.1.

In this excursion, we discuss smoothness properties of the robustness measures $\mathcal{R}_3^{b,\beta}$ and $\mathcal{R}_4^{b,\beta}$. In Example 14 we show that $\mathcal{R}_3^{b,\beta}$ often exhibits large curvatures or even non-smoothness in x, creating a challenge for approximating this robustness measure using surrogate models. We therefore use the quantile reformulation $\mathcal{R}_3^{b,\beta}$ over the probabilistic constraints $\mathcal{R}_3^{b,\beta}$.

Example 14 (Non-smoothness of $\mathcal{R}_3^{b,\beta}$). *Let us consider the two constraints*

$$\begin{aligned} c_1(x,\theta) &= exp\left(\frac{x}{2}\right) - 16(x-2)^2\theta^2 + x - 1, \\ c_2(x,\theta) &= 30x + \theta, \end{aligned} \tag{D.1}$$

which result in the two robust constraints

$$\begin{aligned} \mathcal{R}_3^{c_1,\beta}(x) &= \mathbb{E}\left[\mathbf{1}(c_1(x,\theta) \geq 0)\right] - (1-\beta), \\ \mathcal{R}_3^{c_2,\beta}(x) &= \mathbb{E}\left[\mathbf{1}(c_2(x,\theta) \geq 0)\right] - (1-\beta), \end{aligned} \tag{D.2}$$

with $\theta \sim \mathcal{N}(0,1)$ and $\beta = 0.9$. We compute the sample average estimator using 1000 *samples and plot the robustness measures $\mathcal{R}_3^{c_1,\beta}$ (top left) and $\mathcal{R}_3^{c_2,\beta}(x)$ (bottom left) in Fig. D.1. Besides the sample noise we observe that the response surface of $\mathcal{R}_3^{c_1,\beta}$ has kinks at $x \approx 0$, $x \approx 1.5$ and $x \approx 2.5$ which violate the smoothness assumptions on the constraints; for an in depths discussion about smoothness properties of probability distributions we refer to [178, 309, 310]. Apart from the kinks, even in cases where $\mathcal{R}_3^{c,\beta}$ is arbitrarily smooth, cf. $\mathcal{R}_3^{c_2,\beta}$, it may be a close approximation to a discontinuous step function. The quantile formulations of the probabilistic constraints, $\mathcal{R}_4^{c_1,\beta}$ (top right) and $\mathcal{R}_4^{c_2,\beta}$ (bottom right) in Fig. D.1, on the other hand exhibit smooth behavior.* ◇

To approximate the quantile function $\mathcal{R}_4^{b,\beta}$ we can not rely on the standard MC estimator for approximating $\mathcal{R}_4^{b,\beta}$ anymore. Instead, we follow [338] and use the order statistic $b_{1:N}^x \leq \cdots \leq b_{N:N}^x$, $b_{i:N}^x \in \{b(x,\theta_i)\}_{i=1}^N$ to compute an approximation $b_{\bar{\beta}:N}^x$ of the

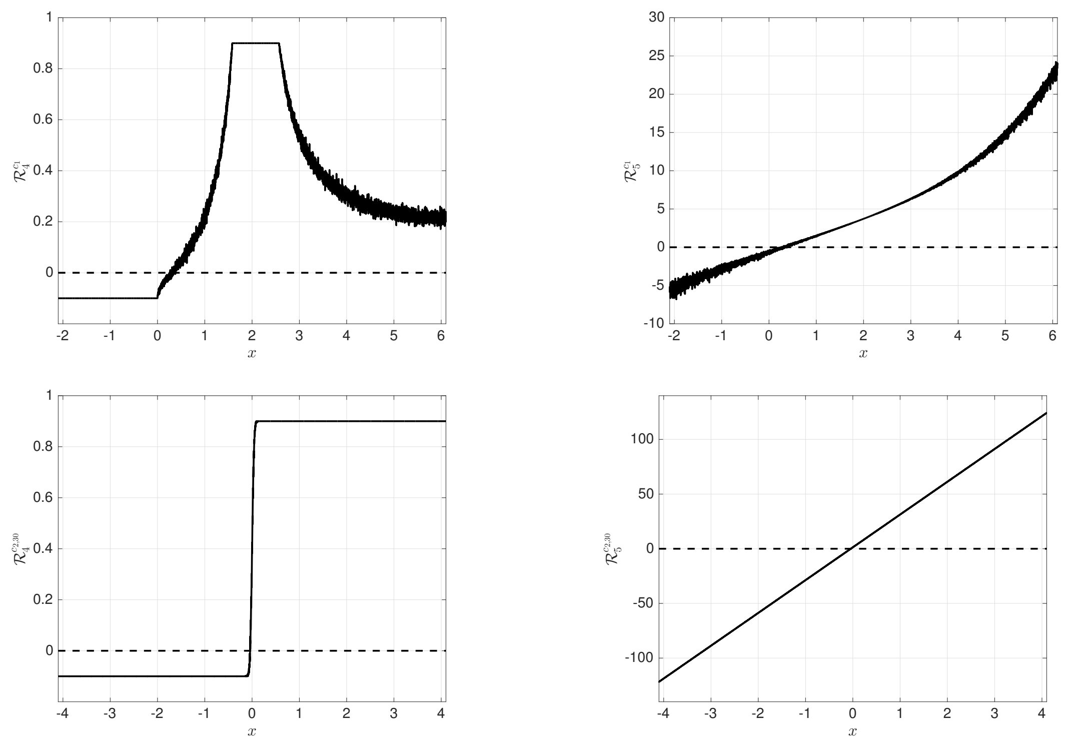


Figure D.1: Sample approximation of $\mathcal{R}_3^{c_1,0.9}$ (upper left) and $\mathcal{R}_4^{c_1,0.9}$ (upper right) based on resampling 1000 samples at each x. The thresholds 0 are plotted as dashed lines. The lower plots show $\mathcal{R}_3^{c_2,0.9}$ (left) and $\mathcal{R}_4^{c_2,0.9}$ (right).

quantile $b_\beta(x)$. More specifically we choose the standard estimator $b^x_{\bar\beta:N} \approx b_\beta(x)$ with

$$\bar\beta = \begin{cases} N\beta, & \text{if } N\beta \text{ is an integer and } \beta < 0.5,\\ N\beta + 1, & \text{if } N\beta \text{ is an integer and } \beta > 0.5,\\ \frac{N}{2} + \mathbf{1}(U \le 0), & \text{if } N\beta \text{ is an integer and } \beta = 0.5,\\ \lfloor N\beta \rfloor + 1, & \text{if } N\beta \text{ is not an integer,} \end{cases}$$

and $U \sim \mathcal{U}[0,1]$, yielding

$$b_\beta(x) = R_3^{b,\beta}(x) + \varepsilon_x = b^x_{\bar\beta:N} + \varepsilon_x. \tag{D.3}$$

Since the order statistic satisfies

$$\mu_b[b^x_{l:N} \le b_\beta(x) \le b^x_{u:N}] \ge \sum_{i=l}^{u-1} \binom{N}{i} \beta^i (1-\beta)^{N-i} =: \pi(l,u,N,\beta),$$

we use it to define a highly probable confidence interval $\left[b^k_{l:N}, b^k_{u:N}\right]$; see [94]. In the same way as for the sample averages we obtain a highly probable upper bound $\bar\varepsilon_x$ on ε_x by choosing

$$\bar\varepsilon_x := \max\left\{ b^x_{\bar\beta:N} - b^x_{(\bar\beta-i):N}, b^x_{(\bar\beta+i):N} - b^x_{\bar\beta:N} \right\},$$

for an $i \in \{1, \dots, N\}$ such that $\pi\left(\bar{\beta} - i, \bar{\beta} + i, N, \beta\right) \geq \nu$ for the confidence level $\nu \in]0, 1[$. We refer to [339] for a detailed discussion about optimal quantile estimators.

D.2 Proof: Variance of Gaussian process mean estimator

Referenced in Lemma 21.

Proof.

$$\mathbb{V}[\mathcal{G}_k^b[R_k^b]] = \mathbb{V}[\mathcal{G}_k^b[\mathbf{x}_k; R^b(\mathbf{X})]] = \mathbb{V}\left[\sum_{j=1}^{N} R^b(\mathbf{x}_j) \sum_{i=1}^{N} \mathbf{k}_{\mathbf{x}_k \mathbf{x}_i} (\mathbf{k}_{\mathbf{x}_i \mathbf{x}_j} + \delta_{ij}\sigma_i^2)^{-\tilde{1}}\right]$$

$$\begin{aligned}
&(\text{Using: } \mathbb{V}[\sum_{i=1}^{N} a_i X_i] = \sum_{i=1}^{N} a_i^2 \mathbb{V}[X_i] + 2 \sum_{1 \leq i < j \leq N} a_i a_j \mathrm{Cov}[X_i, X_j]) \\
&= \sum_{j=1}^{N} \mathbb{V}[R^b(\mathbf{x}_j)] \Bigg(\sum_{i=1}^{N} \mathbf{k}_{\mathbf{x}_k \mathbf{x}_i} (\mathbf{k}_{\mathbf{x}_i \mathbf{x}_j} + \delta_{ij}\sigma_i^2)^{-\tilde{1}} \Bigg)^2 \\
&+ 2 \sum_{1 \leq j < k \leq N} \Bigg[\Bigg(\sum_{i=1}^{N} \mathbf{k}_{\mathbf{x}_k \mathbf{x}_i} (\mathbf{k}_{\mathbf{x}_i \mathbf{x}_j} + \delta_{ij}\sigma_i^2)^{-\tilde{1}} \Bigg) \\
&\Bigg(\sum_{i=1}^{N} \mathbf{k}_{\mathbf{x}_k \mathbf{x}_i} (\mathbf{k}_{\mathbf{x}_i \mathbf{x}_k} + \delta_{ik}\sigma_i^2)^{-\tilde{1}} \Bigg) \mathrm{Cov}[R^b(\mathbf{x}_j), R^b(\mathbf{x}_k)] \Bigg] \\
&(\text{Using } \mathrm{Cov}[R^b(\mathbf{x}_j), R^b(\mathbf{x}_k)] = 0 \text{ for } i \neq j) \\
&= \sum_{j=1}^{N} \mathbb{V}[R^b(\mathbf{x}_j)] \Bigg(\sum_{i=1}^{N} \mathbf{k}_{\mathbf{x}_k \mathbf{x}_i} (\mathbf{k}_{\mathbf{x}_i \mathbf{x}_j} + \delta_{ij}\sigma_i^2)^{-\tilde{1}} \Bigg)^2 \\
&\approx \sum_{j=1}^{N} \left(\frac{\bar{\varepsilon}_{\mathbf{i}}^{\mathbf{b}}}{t_\nu} \right)^2 \Bigg(\sum_{i=1}^{N} \mathbf{k}_{\mathbf{x}_k \mathbf{x}_i} (\mathbf{k}_{\mathbf{x}_i \mathbf{x}_j} + \delta_{ij}\sigma_i^2)^{-\tilde{1}} \Bigg)^2 \\
&(\text{Writing it in matrix notation}) \\
&= \sum_{i=1}^{N} \left(\frac{\bar{\varepsilon}_{\mathbf{i}}^{\mathbf{b}}}{t_\nu} \right)^2 \Bigg(\sum_{j=1}^{N} \mathbf{k}_{x_k x_j} ((K_{\mathbf{XX}} + \mathbf{N})^{-1})_{[i,j]} \Bigg)^2 .
\end{aligned} \tag{D.4}$$

□

D.3 Proof: Covariance of Gaussian process mean estimator and sampling estimator

Referenced in Lemma 22.

Proof.

$$\begin{aligned}
\mathbb{C}\mathrm{ov}[\mathcal{G}_k^b[R_k^b], R_k^b] &= \mathbb{C}\mathrm{ov}[\mathcal{G}_k^b[\mathbf{x}_k; R_k^b(\mathbf{X})], R^b(\mathbf{x}_k)] \\
&= \mathbb{C}\mathrm{ov}[\sum_{j=1}^{N} R^b(\mathbf{x}_j) \sum_{i=1}^{N} \mathbf{k}_{\mathbf{x}_k\mathbf{x}_i}(\mathbf{k}_{\mathbf{x}_i\mathbf{x}_j} + \delta_{ij}\sigma_i^2)^{-\tilde{1}}, R^b(\mathbf{x}_k)] \\
&= \sum_{j=1}^{N} \Big(\sum_{i=1}^{N} \mathbf{k}_{\mathbf{x}_k\mathbf{x}_i}(\mathbf{k}_{\mathbf{x}_i\mathbf{x}_j} + \delta_{ij}\sigma_i^2)^{-\tilde{1}} \Big) \mathbb{C}\mathrm{ov}[R^b(\mathbf{x}_j), R^b(\mathbf{x}_k)] \\
&\text{(uncorrelated up to index } N-1 \text{ since } \mathbf{x}_N = \mathbf{x}_k) \\
&= \Big(\sum_{i=1}^{N} \mathbf{k}_{\mathbf{x}_k\mathbf{x}_i}(\mathbf{k}_{\mathbf{x}_i\mathbf{x}_N} + \delta_{iN}\sigma_i^2)^{-\tilde{1}} \Big) \mathbb{C}\mathrm{ov}[R^b(\mathbf{x}_N), R^b(\mathbf{x}_k)] \\
&= \mathbb{V}[R^b(\mathbf{x}_k)] \sum_{i=1}^{N} \mathbf{k}_{\mathbf{x}_k\mathbf{x}_i}(\mathbf{k}_{\mathbf{x}_i\mathbf{x}_N} + \delta_{iN}\sigma_i^2)^{-\tilde{1}} \\
&\approx (\frac{\bar{\varepsilon}_k^b}{t_\nu})^2 \sum_{i=1}^{N} \mathbf{k}_{\mathbf{x}_k\mathbf{x}_i}(\mathbf{k}_{\mathbf{x}_i\mathbf{x}_N} + \delta_{iN}\sigma_i^2)^{-\tilde{1}} \\
&\text{(Writing it in matrix notation)} \\
&= (\frac{\bar{\varepsilon}_k^b}{t_\nu})^2 \sum_{j=1}^{N} \mathbf{k}_{x_k x_j}((K_{\mathbf{XX}} + \mathbf{N})^{-1})_{[k,j]}.
\end{aligned} \tag{D.5}$$

□

D.4 Hock-Schittkowski-collection

Referenced in Chap. 16, Section 17.1 and Section 17.2.

This section holds the test problem we consider from the Hock-Schittkowski-Collection of optimization benchmarks ([157], [281], [282]). Those problems are also available in the CUTEst optimization framework [144].

TP29

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = x_1 x_2 x_3 \\
\text{s.t.} \quad & c_1 : -48 + x_1^2 + 2x_2 + 4x_3^2 \leq 0
\end{aligned} \tag{D.6}$$

TP43

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4 \\
\text{s.t.} \quad & c_1 : x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8 \leq 0 \\
& c_2 : x_1^2 + 2x_2^2 + x_2^2 + 2x_4^2 - x_1 - x_4 - 10 \leq 0 \\
& c_3 : 2x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 - 5 \leq 0
\end{aligned} \tag{D.7}$$

TP100

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + \\
& \qquad 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \\
\text{s.t.} \quad & c_1 : 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \leq 0 \\
& c_2 : 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \leq 0 \\
& c_3 : 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196 \leq 0 \\
& c_4 : 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0
\end{aligned} \tag{D.8}$$

TP113

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 \\
& \qquad + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 \\
& \qquad + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_10 - 7)^2 + 45 \\
\text{s.t.} \quad & c_1 : 4x_1 + 5x_2 - 3x_7 + 9x_8 - 105 \leq 0 \\
& c_2 : 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \\
& c_3 : -8x_1 + 2x_2 + 5x_9 - 2x_10 - 12 \leq 0 \\
& c_4 : 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \\
& c_5 : 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \\
& c_6 : 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0 \\
& c_7 : x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0 \\
& c_8 : -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_10 \leq 0
\end{aligned} \tag{D.9}$$

TP227

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2 \\
\text{s.t.} \quad & c_1 : x_1^2 - x_2 \leq 0 \\
& c_2 : x_1 + x_2^2 \leq 0
\end{aligned} \tag{D.10}$$

TP228

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = x_1^2 + x_2 \\
\text{s.t.} \quad & c_1 : x_1 + x_2 - 1 \leq 0 \\
& c_2 : -x_1^2 - x_2^2 - 9 \leq 0
\end{aligned} \tag{D.11}$$

TP268

$$\mathbf{V} = \begin{pmatrix} -74 & 80 & 18 & -11 & -4 \\ -14 & -69 & 21 & 28 & 0 \\ -66 & -72 & -5 & 7 & 1 \\ -12 & -66 & -30 & -24 & 3 \\ 3 & 8 & -7 & -4 & 1 \\ 4 & -12 & 4 & 4 & 0 \end{pmatrix}, \mathbf{v} = \begin{pmatrix} 51 \\ -61 \\ -56 \\ -69 \\ 10 \\ -12 \end{pmatrix}, \mathbf{W} = \mathbf{V}^T\mathbf{V}, \mathbf{w} = \mathbf{v}^T\mathbf{V} \tag{D.12}$$

$$\begin{aligned} \min \quad & f(\mathbf{x}) = 14463 + \sum_{i=1}^{5} \left[x_i (\sum_{j=1}^{5} (W_{ij} x_j) - 2w_i) \right] \\ \text{s.t.} \quad & c_1 : x_1 + x_2 + x_3 + x_4 + x_5 - 5 \leq 0 \\ & c_2 : -10x_1 - 10x_2 + 3x_3 - 5x_4 - 4x_5 + 20 \leq 0 \\ & c_3 : 8x_1 - x_2 + 2x_3 + 5x_4 - 3x_5 - 40 \leq 0 \\ & c_4 : -8x_1 + x_2 - 2x_3 - 5x_4 + 3x_5 + 11 \leq 0 \\ & c_5 : 4x_1 + 2x_2 - 3x_3 + 5x_4 - x_5 - 30 \leq 0 \end{aligned} \tag{D.13}$$

TP285

$$\mathbf{A} \in \mathbb{Z}^{10\times 15}, \mathbf{a} \in \mathbb{Z}^{15}, \mathbf{b} \in \mathbb{Z}^{10} \tag{D.14}$$

$$\begin{aligned} \min \quad & f(\mathbf{x}) = \sum_{i=1}^{15} (-a_i x_i) \\ \text{s.t.} \quad & c_i : -b_i + \sum_{j=1}^{10} \sum_{k=1}^{15} (A_{jk} x_k^2) \leq 0 \end{aligned} \tag{D.15}$$

D.5 Approximate Gaussian processes in SNOWPAC

Referenced in Section 17.2.

Results for (16.3):

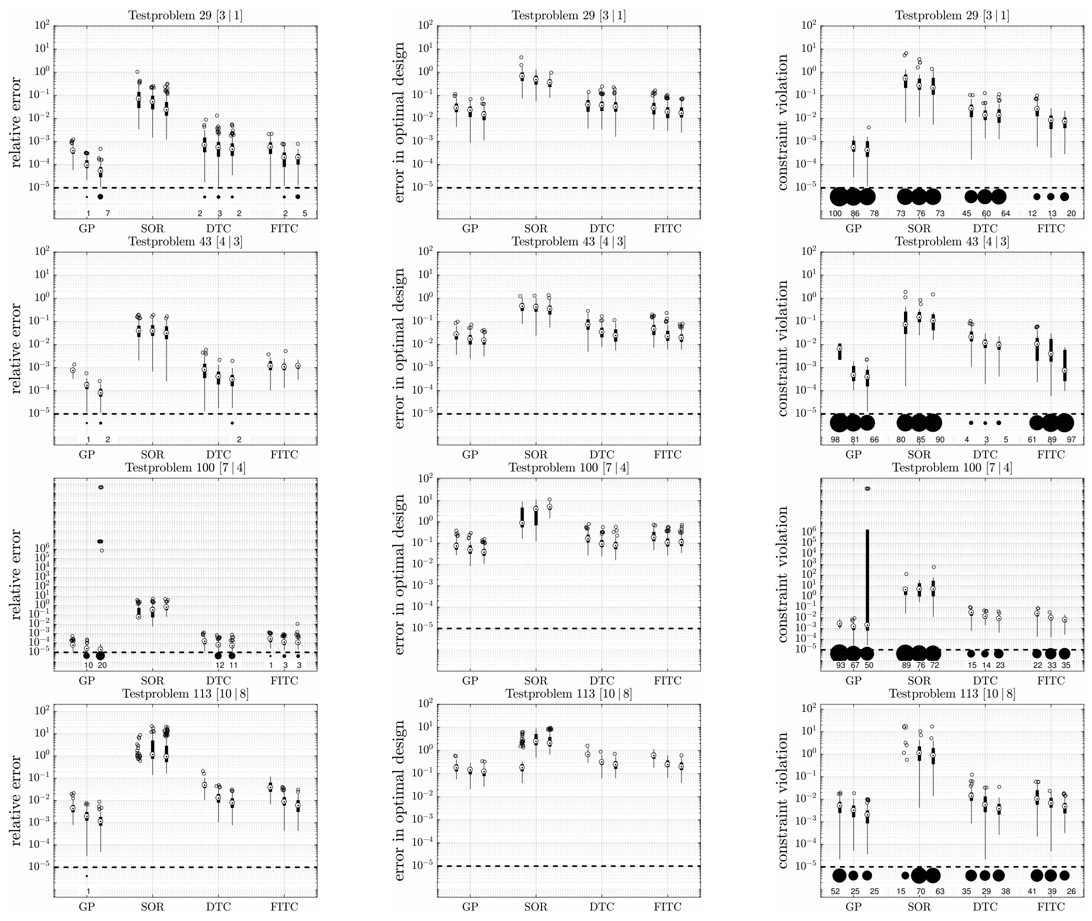


Figure D.2: SNOWPAC Schittkowski benchmark for test problem TP = 29, TP = 43, TP = 100 and TP = 113 for relative error of the objective function, absolute error in optimal design and the constraint violations from left to right for optimization problem (16.3).

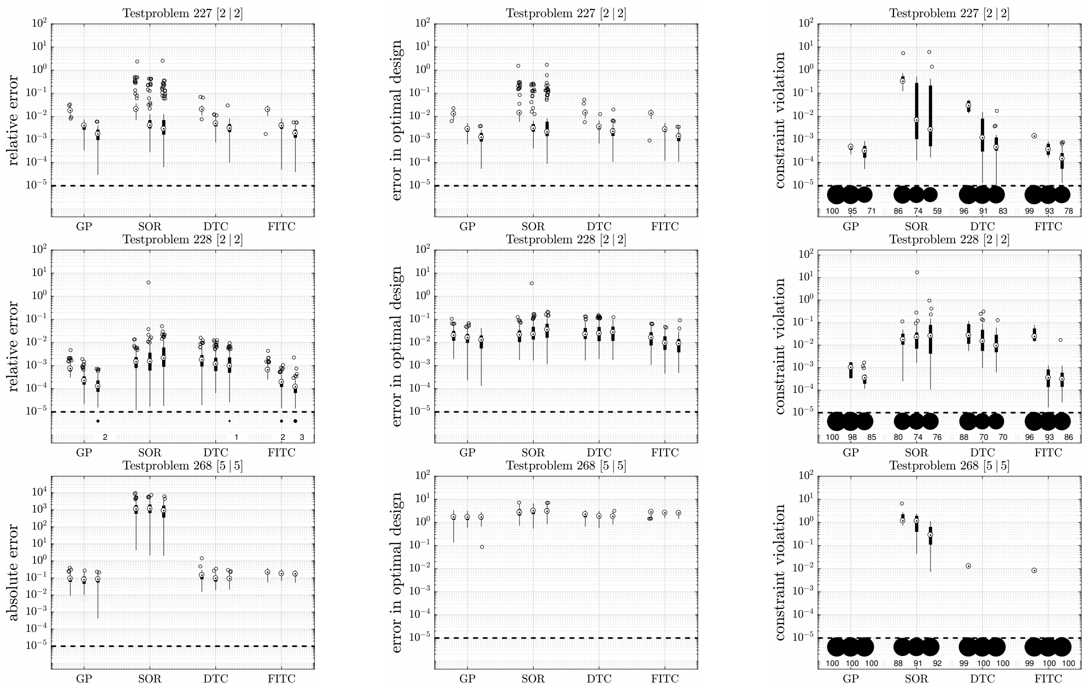


Figure D.3: SNOWPAC Schittkowski benchmark for test problem TP = 227, TP = 228 and TP = 268 for relative error of the objective function, absolute error in optimal design and the constraint violations from left to right for optimization problem (16.3).

D.6 Mixed integer optimization

D.6.1 Box-constrained problems

Referenced in Section 17.3.

1. **Sphere Problem [333]:** This is a simple convex problem. We use this problem with various number of design parameters. It is turned mixed-integer by imposing integer constraint on the first half number of design parameters.

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2, \\ s.t. \quad & -7 \le x_i \le 7, \quad i = 1, ..., n \\ & x_j \in \mathbb{Z}, \quad j = 1, ..., \frac{n}{2} \\ & x_k \in \mathbb{R}, \quad k = \frac{n}{2} + 1, ..., n. \end{aligned} \tag{D.16}$$

This optimization problem has one global minimum at $\mathbf{x}^* = (0, ..., 0)$ with $f(\mathbf{x}^*) = 0$.

2. **Ackley's Function [10, 298, 333]:** This is a multi-modal function. We use eight dimensions with three integer design parameters and five real design parameters.

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = -20exp\left[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right] - exp\left[\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right] + 20 + e \\ s.t. \quad & -7 \le x_i \le 7, \quad i = 1, ..., 8 \\ & x_j \in \mathbb{Z}, \quad j = 1, 2, 3 \\ & x_k \in \mathbb{R}, \quad k = 4, ..., n. \end{aligned} \tag{D.17}$$

where $n = 8$. This optimization problem has many local minimums, with global minimum at $\mathbf{x}^* = (0, ..., 0)$ with $f(\mathbf{x}^*) = 0$.

3. **Weighted De Jong's Function [333]:** This is a uni-modal function. It is also called *hyper-ellipsoid function.* Weighting of axis makes this optimization problem a little difficult to solve. We use five dimensions and turn it into mixed-integer by imposing integer constraint on the first three design parameters.

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2, \\ s.t. \quad & -7 \le x_i \le 7, \quad i = 1, ..., n \\ & x_j \in \mathbb{Z}, \quad j = 1, 2, 3 \\ & x_k \in \mathbb{R}, \quad k = 4, , , , n. \end{aligned} \tag{D.18}$$

where $n = 5$. This optimization problem has one global minimum at $\mathbf{x}^* = (0, ..., 0)$ with $f(\mathbf{x}^*) = 0$.

4. **Bohachevsky Problem 1 (BF1) [10, 48]:** This is a multi-modal function.We impose integer constraint on first variable.

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = x_i^2 + 2x_2^2 - 0.3cos(3\pi x_1) - 0.4cos(4\pi x_2) + 0.7 \\ s.t. \quad & -7 \leq x_i \leq 7, \quad i = 1, 2 \\ & x_1 \in \mathbb{Z} \quad x_2 \in \mathbb{R}. \end{aligned} \tag{D.19}$$

   This optimization problem has many local minimums, with global minimum at $\mathbf{x}^* = (0, 0)$ with $f(\mathbf{x}^*) = 0$.

5. **Bohachevsky Problem 2 (BF2) [10, 48]:** This is a multi-modal function. We impose integer constraint on first variable

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = x_i^2 + 2x_2^2 - 0.3cos(3\pi x_1)cos(4\pi x_2) + 0.3 \\ s.t. \quad & -7 \leq x_i \leq 7, \quad i = 1, 2 \\ & x_1 \in \mathbb{Z} \quad x_2 \in \mathbb{R}. \end{aligned} \tag{D.20}$$

   This optimization problem has many local minimums, with global minimum at $\mathbf{x}^* = (0, 0)$ with $f(\mathbf{x}^*) = 0$.

6. **Griewank's Function [333]:** This is a multi-modal function. We use ten dimensions with five integer design parameters and five real design parameters.

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \\ s.t. \quad & -7 \leq x_i \leq 7, \quad i = 1, ..., 8 \\ & x_j \in \mathbb{Z}, \quad j = 1, , .., 5 \\ & x_k \in \mathbb{R}, \quad k = 4, ..., 8. \end{aligned} \tag{D.21}$$

   where $n = 10$. This optimization problem has many local minimums, with global minimum at $\mathbf{x}^* = (0, ..., 0)$ with $f(\mathbf{x}^*) = 0$.

D.6.2 Hyperparameter list

Referenced in Section 17.3.

Hyperparameter(λ)	**Type**	**Scaling**	**Range**	**Initial Point**
Number of training Epochs	Integer	5λ	[1 , 10]	2
Number of hidden nodes	Integer	2^{λ}	[4 , 10]	5
Learning rate of SGD	Continuous	10^{λ}	[-3 , $log_{10}0.2$]	-2
Momentum of SGD	Continuous	$\lambda/10$	[6 , 9]	8
Mean of Gaussian initialization	Continuous	10^{λ}	[-5 , -2]	-4
Mean of Gaussian initialization	Continuous	10^{λ}	[-5 , -2]	-4

Table D.1: List of hyperparameters for second category of 6-MLP problems with range and initial point.

Hyperparameter(λ)	**Type**	**Scaling**	**Range**	**Initial Point**
Depth of first Conv layer	Integer	2^{λ}	[1 , 7]	2
Depth of second Conv layer	Integer	2^{λ}	[1 , 7]	2
Number of hidden nodes in first FC layer	Integer	2^{λ}	[1 , 10]	5
Number of hidden nodes in second FC layer	Integer	2^{λ}	[1 , 10]	5
Learning rate of SGD	Continuous	10^{λ}	[-3 , $log_{10}0.3$]	-1
Momentum of SGD	Continuous	$\lambda/10$	[6 , 9]	8
Weight decay rate	Continuous	10^{λ}	[-5 , -2]	-4
Learning rate decay	Continuous	10^{λ}	[-5 , -2]	-4

Table D.2: List of hyperparameters for second category of 8-CNN problems with range and initial point.

Hyperparameter(λ)	**Type**	**Scaling**	**Range**	**Initial Point**
Mini-batch size	Integer	2^λ	[4 , 8]	5
Depth of first Conv layer	Integer	2^λ	[1 , 7]	2
Depth of second Conv layer	Integer	2^λ	[1 , 7]	2
Number of hidden nodes in first FC layer	Integer	2^λ	[1 , 10]	5
Number of hidden nodes in second FC layer	Integer	2^λ	[1 , 10]	5
Learning rate of SGD	Continuous	10^λ	[-3 , $log_{10}0.3$]	-1
Momentum of SGD	Continuous	$\lambda/10$	[6 , 9]	8
Weight decay rate	Continuous	10^λ	[-5 , -2]	-4
Learning rate decay	Continuous	10^λ	[-5 , -2]	-4
α leaky ReLU in first FC Layer	Continuous	$\lambda/10$	[0 , 5]	0.1
α leaky ReLU in second FC Layer	Continuous	$\lambda/10$	[0 , 5]	0.1
STD of Gaussian initialization for first FC layer	Continuous	$\lambda/10$	[0 , 5]	0.1
STD of Gaussian initialization for second FC layer	Continuous	$\lambda/10$	[0 , 5]	0.1
STD of Gaussian initialization for first Conv layer	Continuous	$\lambda/10$	[0 , 5]	0.1
STD of Gaussian initialization for second Conv layer	Continuous	$\lambda/10$	[0 , 5]	0.1

Table D.3: List of hyperparameters for second category of 15-CNN problems with range and initial point.

Hyperparameter(λ)	**Type**	**Scaling**	**Range**	**Initial Point**
Mini-batch size	Integer	2^λ	[4 , 8]	5
Depth of first Conv layer	Integer	2^λ	[1 , 7]	2
Depth of second Conv layer	Integer	2^λ	[1 , 7]	2
Number of hidden nodes in first FC layer	Integer	2^λ	[1 , 10]	5
Number of hidden nodes in second FC layer	Integer	2^λ	[1 , 10]	5
Learning rate of SGD	Continuous	10^λ	[-3 , $log_{10}0.3$]	-1
Momentum of SGD	Continuous	$\lambda/10$	[6 , 9]	8
Weight decay rate	Continuous	10^λ	[-5 , -2]	-4
Learning rate decay	Continuous	10^λ	[-5 , -2]	-4
α leaky ReLU in first FC Layer	Continuous	$\lambda/10$	[0 , 5]	0.1
α leaky ReLU in second FC Layer	Continuous	$\lambda/10$	[0 , 5]	0.1
STD of Gaussian initialization for first FC layer	Continuous	$\lambda/10$	[0 , 5]	0.1
STD of Gaussian initialization for second FC layer	Continuous	$\lambda/10$	[0 , 5]	0.1
STD of Gaussian initialization for first Conv layer	Continuous	$\lambda/10$	[0 , 5]	0.1
STD of Gaussian initialization for second Conv layer	Continuous	$\lambda/10$	[0 , 5]	0.1
Dropout rate for first FC layer	Continuous	λ	[0.00 , 0.80]	0.5
Dropout rate for second FC layer	Continuous	λ	[0.00 , 0.80]	0.5
Dropout rate for first Conv layer	Continuous	λ	[0.00 , 0.80]	0.5
Dropout rate for second Conv layer	Continuous	λ	[0.00 , 0.80]	0.5

Table D.4: List of hyperparameters for second category of 19-CNN problems with range and initial point.

	6-MLP	**8-CNN**	**15-CNN**	**19-CNN**
Category 1	70	200	200	200
Category 2	5	5	5	5

Table D.5: Starting half box-dimension for SNOWPAC for all the hyperparameter optimization problems.

E Multilevel Monte Carlo for optimization under uncertainty

E.1 Problem 18 OUU: input file example

Referenced in Section 19.2.

```
#@ *: DakotaConfig=HAVE_NOWPAC

environment,
    tabular_data
        tabular_data_file = 'dakota_problem18_ouu_scalarization.dat'
    method_pointer = 'OPTIM'

###########################
# begin opt specification #
###########################
method,
    id_method = 'OPTIM'
    model_pointer = 'OPTIM_M'
    snowpac
      seed = 25041981
      max_iterations = 100
      max_function_evaluations = 100
      trust_region
        initial_size = 0.05
        minimum_size = 1.0e-6
        contract_threshold = 0.25
        expand_threshold   = 0.75
        contraction_factor = 0.50
        expansion_factor   = 1.50
      output silent

model,
    id_model = 'OPTIM_M'
    nested
      variables_pointer  = 'OPTIM_V'
      sub_method_pointer = 'UQ'
      responses_pointer  = 'OPTIM_R'
      primary_response_mapping   = 1. 3. 0. 0.
      secondary_response_mapping = 0. 0. 1. 0.

variables,
    id_variables = 'OPTIM_V'
    continuous_design = 1
```

```
      initial_point        0.25
      upper_bounds       6
      lower_bounds       0
      descriptors        'x'

responses,
    id_responses = 'OPTIM_R'
    objective_functions = 1
    nonlinear_inequality_constraints = 1
    nonlinear_inequality_lower_bounds = -1.e+50
    nonlinear_inequality_upper_bounds = 0
    no_gradients
    no_hessians

##########################
# begin UQ specification #
##########################
method,
    id_method = 'UQ'
    model_pointer = 'HIERARCH'
      multilevel_sampling
      pilot_samples = 10000 1000 100 50
      final_moments standard
      max_iterations = 20
      convergence_tolerance 1.617509453025866e-05
        convergence_tolerance_type absolute
        convergence_tolerance_target variance_constraint
      sample_type random
      output silent
      allocation_target scalarization
        optimization
      qoi_aggregation max

model,
    id_model = 'HIERARCH'
    variables_pointer = 'UQ_V'
    responses_pointer = 'UQ_R'
    surrogate hierarchical
      ordered_model_fidelities = 'MLModel'

model,
    id_model = 'MLModel'
    variables_pointer = 'UQ_V'
    interface_pointer = 'UQ_I'
    responses_pointer = 'UQ_R'
    simulation
        solution_level_control = 'Af'
        solution_level_cost = 0.01 1.0 0.1 0.001

 variables,
    id_variables = 'UQ_V'
    continuous_design = 1
    uniform_uncertain = 1
      lower_bounds      =  -0.5
      upper_bounds      =  0.5
           descriptors       =  'xi'
    discrete_state_set  real = 2
             num_set_values = 4 1
        initial_state = 1.5  0.0
        set_values = -6 1.0  1.1  1.5
                              0.0
        descriptors = 'Af'  'Ac'

interface ,
    id_interface = 'UQ_I'
    direct
        analysis_driver = 'problem18'
        deactivate
                restart_file
                evaluation_cache

responses ,
    id_responses = 'UQ_R'
    response_functions = 2
    no_gradients
    no_hessians
```

Listing E.1: Dakota input file for OUU using MLMC sampling of Problem 18 targeting the scalarization case of Section 19.2.

E.2 Rosenbrock OUU: input file example

Referenced in Section 19.3.

```
#@ *: DakotaConfig=HAVE_NOWPAC

environment,
    tabular_data
        tabular_data_file = 'dakota_problem18_ouu_scalarization.dat'
    method_pointer = 'OPTIM'

###########################
# begin opt specification #
###########################
method,
    id_method = 'OPTIM'
    model_pointer = 'OPTIM_M'
    snowpac
      seed = 25041981
      max_iterations = 100
      max_function_evaluations = 100
      trust_region
        initial_size = 0.05
        minimum_size = 1.0e-6
        contract_threshold = 0.25
        expand_threshold   = 0.75
        contraction_factor = 0.50
        expansion_factor   = 1.50
      output silent

model,
    id_model = 'OPTIM_M'
    nested
      variables_pointer   = 'OPTIM_V'
      sub_method_pointer = 'UQ'
      responses_pointer   = 'OPTIM_R'
      primary_response_mapping    = 1. 3. 0. 0.
      secondary_response_mapping = 0. 0. 1. 0.

variables,
    id_variables = 'OPTIM_V'
    continuous_design = 1
      initial_point       0.25
      upper_bounds      6
      lower_bounds      0
      descriptors       'x'

responses,
    id_responses = 'OPTIM_R'
    objective_functions = 1
    nonlinear_inequality_constraints = 1
    nonlinear_inequality_lower_bounds = -1.e+50
    nonlinear_inequality_upper_bounds = 0
    no_gradients
    no_hessians

##########################
# begin UQ specification #
##########################
method,
    id_method = 'UQ'
    model_pointer = 'HIERARCH'
      multilevel_sampling
      pilot_samples = 10000 1000 100 50
      final_moments standard
      max_iterations = 20
      convergence_tolerance 1.617509453025866e-05
        convergence_tolerance_type absolute
        convergence_tolerance_target variance_constraint
      sample_type random
      output silent
      allocation_target scalarization
        optimization
      qoi_aggregation max

model,
    id_model = 'HIERARCH'
    variables_pointer = 'UQ_V'
    responses_pointer = 'UQ_R'
    surrogate hierarchical
      ordered_model_fidelities = 'MLModel'

model,
    id_model = 'MLModel'
    variables_pointer = 'UQ_V'
    interface_pointer = 'UQ_I'
    responses_pointer = 'UQ_R'
    simulation
        solution_level_control = 'Af'
        solution_level_cost = 0.01 1.0 0.1 0.001

 variables,
    id_variables = 'UQ_V'
    continuous_design = 1
    uniform_uncertain = 1
      lower_bounds      =  -0.5
      upper_bounds      =  0.5
           descriptors       =  'xi'
    discrete_state_set real = 2
             num_set_values = 4 1
        initial_state = 1.5 0.0
        set_values = -6 1.0 1.1 1.5
                          0.0
        descriptors = 'Af' 'Ac'

interface,
    id_interface = 'UQ_I'
    direct
        analysis_driver = 'problem18'
        deactivate
                restart_file
                evaluation_cache

responses ,
    id_responses = 'UQ_R'
    response_functions = 2
    no_gradients
    no_hessians
```

Listing E.2: Dakota input file for OUU using MLMC sampling of Rosenbrock targeting the scalarization case of Section 19.3.

E.3 Wind OUU: input file example

Referenced in Section 19.4.3.

```
#@ *: DakotaConfig=HAVE_NOWPAC

environment ,
    tabular_data
    method_pointer = 'OPTIM'

###########################
# begin opt specification #
###########################
method,
    id_method = 'OPTIM'
    model_pointer = 'OPTIM_M'
    snowpac
      seed = 25041981
      max_function_evaluations = 200
      trust_region
        initial_size = 0.15
        minimum_size = 1.0e-6
        contract_threshold = 0.25
        expand_threshold   = 0.75
        contraction_factor = 0.50
        expansion_factor   = 1.50
      output debug

model,
    id_model = 'OPTIM_M'
    nested
      variables_pointer  = 'OPTIM_V'
      sub_method_pointer = 'UQ'
      responses_pointer  = 'OPTIM_R'
      primary_response_mapping   = 1. 3.

variables ,
    id_variables = 'OPTIM_V'
    continuous_design = 9
      initial_point      0*0.
      upper_bounds       9*30.
      lower_bounds      9*-30.
      descriptors        'yaw_angle_1'     'yaw_angle_2'  'yaw_angle_3'
      'yaw_angle_4'  'yaw_angle_5'  'yaw_angle_6'
      'yaw_angle_7'  'yaw_angle_8'  'yaw_angle_9'


responses ,
    id_responses = 'OPTIM_R'
    objective_functions = 1
    no_gradients
    no_hessians

```

```
###########################
# begin UQ specification #
###########################
method,
    id_method = 'UQ'
    model_pointer = 'HIERARCH'
      multilevel_sampling
      pilot_samples = 5000 100 10
      final_moments standard
      max_iterations = 20
      convergence_tolerance 1000
        convergence_tolerance_type relative
        convergence_tolerance_target cost_constraint
      fixed_seed
      output debug
      allocation_target scalarization
        optimization
      qoi_aggregation max

model,
    id_model = 'HIERARCH'
    variables_pointer = 'UQ_V'
    responses_pointer = 'UQ_R'
    surrogate hierarchical
      ordered_model_fidelities = 'MLModel'

model,
    id_model = 'MLModel'
    variables_pointer = 'UQ_V'
    interface_pointer = 'UQ_I'
    responses_pointer = 'UQ_R'
    simulation
        solution_level_control = 'model'
        solution_level_cost = 0.0025 1.0 0.12

variables,
    id_variables = 'UQ_V'
    continuous_design = 5
    normal_uncertain = 2
        descriptors  =   'HH_vel' 'wind_angle'
        means =    7.5       0
        std_deviations =   1     5
    discrete_state_set string = 1
        num_set_values = 3
        initial_state = 'coarse'
        set_values = 'coarse' 'fine' 'medium'
        descriptors = 'model'

interface,
    id_interface = 'UQ_I'
    fork asynchronous evaluation_concurrency = 1
      analysis_driver = 'interface_DAK2Floris.sh'
      parameters_file = 'params.in'
      results_file = 'results.out'
```

```
        file_tag

responses,
    id_responses = 'UQ_R'
    response_functions = 1
    no_gradients
    no_hessians
```

Listing E.3: Dakota input file for OUU using MLMC sampling of Floris targeting the scalarization case of Section 19.4.