Technical University of Munich
TUM School of Engineering and Design

Probabilistic Modeling and Scientific Machine Learning for Computational Physics and Engineering

Atul Agrawal

Complete reprint of the dissertation approved by the TUM School of Engineering and Design of the Technical University of Munich for the award of the

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

The dissertation was submitted to the Technical University of Munich on 29 August 2024 and accepted by the TUM School of Engineering and Design on 5 February 2025.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Pheadon-Stelios Koutsourelakis. Thank you for your invaluable guidance, support, and the wealth of knowledge you shared with me. Your unwavering dedication to science has been a constant source of motivation. Your perspectives and insights were crucial in helping me navigate complex topics. I am also grateful for your understanding and support during the challenging times of the pandemic. Thank you for pushing me when needed and for your incredible patience throughout this journey. I have learned so much from you. I would also like to extend my sincere gratitude to Prof. Felix Dietrich for agreeing to be a member of the examination committee for this thesis, and to Prof. Rafael Macián-Juan for chairing the committee. I would like to extend my gratitude to my collaborators from the Federal Institute for Materials Research and Testing (BAM) and the Karlsruhe Institute of Technology (KIT). I am also thankful to my collaborators Kislaya Ravi and Prof. Hans J. Bungartz from the Chair of Scientific Computing (TUM) for their valuable contributions. Before beginning my PhD, I had the privilege of working with some of the finest scientists, who inspired me and instilled foundational values essential for becoming a good researcher. I would like to extend my heartfelt thanks to Prof. Dr. D. Roy Mohapatra (Indian Institute of Science, Bangalore), Prof. Dr. Didier Lucor and Prof. Dr. Anne Sergent (University Paris Saclay), and Prof. Dr. Nicolas Möes (Ecole Centrale Nantes) for their guidance.

I was fortunate to have a great cohort of colleagues and friends who made this journey worthwhile. Firstly, I would like to thank all my friends and current or former colleagues at the Professorship of Data-driven Materials Modeling (DDMM): Maximilian Rixner, Sebastian Kaltenbach, Constantin Grigo, Markus Schöberl, Jonas Nitzler, Luca Berardocco, Matthaios Chatzopoulos, Vincent Scholz, Maximilian Stupp and Yaohua Zang. I learned so much from each of you and thoroughly enjoyed our lunchtime discussions and post-work gatherings. Special thanks to Max Rixner for your patience and constant support during the early years of my PhD. Also, thanks Jonas for all the crazy stuff we did together—we still have to make that rock and roll band idea happen! I also extend my gratitude to our group secretary, Sigrid Harnauer, for your kindness and assistance with administrative tasks and navigating bureaucratic hurdles. I would also like to thank my colleagues from the Institute for Computational Mechanics (LNM) for always including me in your get-togethers, seminars, and after-parties. Special thanks to Jonas, Ravi, and Vincent for thoroughly proofreading this thesis. I am also grateful to Niklas Fehn from LNM for the invaluable advice on thesis writing and many interesting discussions during my initial years. Lastly, I want to thank all my friends in and around Munich for sharing countless crazy stories and experiences with me, and my long-time friends from IIT-BHU for being such a strong pillar of support throughout these years.

I could not have completed this journey without the unwavering support of my family. I am deeply grateful to my wife, Nabanita, for being a constant source of strength during the later years of my PhD and for her endless patience. I extend my heartfelt thanks to my parents, Saroj and Pradip Kumar Agrawal, for always believing in me and supporting me. Especially my father, who has been a constant source of inspiration, fostering a deep sense of curiosity within me. I am also thankful to my brother, Ayush Agrawal, for always being there for me. I would like to express my gratitude to all my aunts, uncles, and cousins who have consistently cheered me on throughout these years. A special thanks to my uncle, Rajesh Agrawal, for his constant guidance in both my professional and personal life. Lastly, I dedicate this thesis to the memory of my late grandfather, Dev Karan Agrawal. Words cannot fully capture the impact you had on my life. I hope this thesis would have earned a place on your shelf alongside your hard-bound law books.

Abstract

As the complexity of the scientific problems we want to study increases, machine learning (ML) is helping to automate, accelerate and enhance traditional workflows. This is achieved by combining existing scientific understanding with ML, a relatively new field known as scientific machine learning (SciML). This thesis explores the integration of ML with physics-based modeling to overcome the limitations of traditional computational methods in engineering and science. The work is motivated by the challenges of applying ML in scientific domains, including the scarcity of large labeled datasets, noise in data, difficulties in generalization, and the complexity of coupling ML models with non-differentiable, physics-based simulations. To address these issues, we propose a suite of novel strategies that combines SciML with probabilistic modeling and differentiable physics. These strategies are then used to address challenging computational physics and engineering applications.

Firstly, we propose strategies for turbulence closure modeling. We propose a probabilistic, data-driven closure model for Reynolds-Averaged Navier-Stokes (RANS) simulations. The model integrates a RANS solver with a ML model, enabling end-to-end gradient-based learning through an adjoint-based differentiable solver. The ML model is designed with hard constraints to ensure physical symmetry and invariance, and it employs fully Bayesian learning using sparse, indirect data such as mean velocity and pressure. The closure model incorporates both a parametric component, using neural-network-based tensor basis functions, and a stochastic component for aleatoric model uncertainties. The model is demonstrated to produce accurate probabilistic predictions, even in regions with significant model errors, as evidenced by its performance in the backward-facing step benchmark problem, which involves flow re-circulation and separation.

Next, we develop a holistic optimization framework that combines concrete mixture design and structural simulation, consisting of empirical data, semi-analytical models, physics-based models, and ML techniques. This framework includes a model discovery method using variational Bayes expectation-maximization, which identifies the relationships between design variables and model parameters by making use of noisy and incomplete experimental data and finite element simulations. The framework is applied to the design of a concrete beam, achieving a balance between minimizing environmental impact and meeting performance criteria.

Finally, we propose a novel optimization algorithm recognizing the complexities of optimizing black-box, stochastic, high-dimensional, and computationally expensive physics-based models. This algorithm incorporates techniques such as efficient gradient estimation, multi-fidelity methods, and adaptive Monte Carlo sampling, enabling it to handle complex physics-based models involved in the optimization process. The algorithm is validated against academic benchmark problems and successfully applied to a real-world case of optimizing wind farm layout. We demonstrate that the proposed algorithm outperforms the state-of-the-art solvers in high-dimensional cases.

Zusammenfassung

Mit der zunehmenden Komplexität der zu untersuchenden wissenschaftlichen Probleme trägt maschinelles Lernen (ML) dazu bei, traditionelle Arbeitsabläufe zu automatisieren, zu beschleunigen und zu verbessern. Dies wird durch die Kombination bestehender wissenschaftlicher Erkenntnisse mit ML erreicht, einem relativ neuen Bereich, der als wissenschaftliches maschinelles Lernen (SciML) bekannt ist. Diese Dissertation untersucht die Integration von ML in die physikbasierte Modellierung, um die Grenzen traditioneller rechnergestützter Methoden in Ingenieurwissenschaften und Naturwissenschaften zu überwinden. Die Motivation für diese Arbeit ergibt sich aus den Herausforderungen bei der Anwendung von ML in wissenschaftlichen Bereichen, einschließlich des Mangels an großen vorklassifizierten Datensätzen, Rauschen in den Daten, Schwierigkeiten bei der Generalisierung und der Komplexität der Kopplung von ML-Modellen mit nicht-differenzierbaren, physikbasierten Simulationen. Um diese Probleme zu überwinden, schlagen wir eine Reihe neuartiger Strategien vor, die SciML mit probabilistischer Modellierung und differenzierbaren, physikalischen Systemen kombinieren. Diese Strategien werden anschließend verwendet, um anspruchsvolle Anwendungen in der numerischen Physik und im Ingenieurwesen zu lösen.

Zunächst präsentieren wir Strategien für die Modellieung von Turbulenzschließungen. Wir haben ein probabilistisches, datengetriebenes Schließungsmodell für Reynolds-gemittelte Navier-Stokes (RANS)-Simulationen entwickelt. Das Modell integriert ein Lösungsverfahren für RANS mit einem maschinellen Lernmodell (ML), das ein end-to-end, gradientenbasiertes Lernen durch ein adjungiertes, differenzierbares Lösungsverfahren ermöglicht. Das ML-Modell wird harter Bedingungen entworfen, um physikalische Symmetrie und Invarianz zu gewährleisten, und es verwendet einen vollständigen, bayesianischen Lernansatz unter Verwendung von unvollständigen, indirekten Daten wie mittlerer Geschwindigkeit und Druck. Das Schließungsmodell enthält sowohl eine parametrische Komponente, die auf neuralen Netzwerk-basierten Tensorbasisfunktionen beruht, als auch eine stochastische Komponente zur Berücksichtigung von aleatorischen Modellunsicherheiten. Es wird gezeigt, dass das Modell auch in Bereichen mit erheblichen Modellfehlern genaue, probabilistische Vorhersagen liefert, wie am Beispiel der Strömungsrückführung und -trennung beim rückwärts gerichteten Stufenproblem demonstriert wird.

Als Nächstes entwickeln wir ein ganzheitliches Optimierungsverfahren, das Betonmischungsplanung und Struktursimulation kombiniert und aus empirischen Daten, semi-analytischen Modellen, physikbasierten Modellen und maschinellen Lerntechniken besteht. Dieses Verfahren umfasst eine Modell-Discovery-Methode, die einen bayesschen Erwartungs-Maximierungs Algorithmus verwendet und die Beziehungen zwischen Designvariablen und Modellparametern durch die Nutzung von verrauschten und unvollständigen experimentellen Daten und Finite-Elemente-Simulationen identifiziert. Das Verfahren wird auf das Design eines Betonbalkens angewendet, um ein Gleichgewicht zwischen der Minimierung der Umweltbelastung und der Erfüllung der Leistungskriterien zu erreichen.

Abschließend entwickeln wir einen neuartigen Optimierungsalgorithmus, der die Komplexität der Optimierung von Black-Box-, stochastischen, hochdimensionalen und rechnerisch aufwändigen physikbasierten Modellen berücksichtigt. Dieser Algorithmus integriert Techniken wie effiziente Gradientenabschätzung, Methoden mit unterschiedlicher Genauigkeit und adaptives Monte-Carlo-Sampling, und ermöglicht so die Handhabung komplexer, physikbasierter Modelle im Optimierungsprozess. Der Algorithmus wird an akademischen Benchmark-Problemen validiert und erfolgreich auf das reale Fallbeispiel der Optimierung einer Windpark-Anordnung angewendet. Wir zeigen, dass der entwickelte Algorithmus in hochdimensionalen Fällen den Stand der Technik übertrifft.

Contents

List of Figures

Acronyms

AD	Automatic Differentiation
ARD	Automatic Relevance Determination
BBVI	Blackbox Variational Inference
CFD	Computational Fluid Dynamics
DNN	Deep Neural Network
DNS	Direct Numerical Simulation
DeepONet	Deep Operator Network
ELBO	Evidence Lower Bound
EM	Expectation-Maximization
FEM	Finite Element Method
FFNN	Feed-Forward Neural Network
FNO	Fourier Neural Operator
FVM	Finite Volume Method
GNN	Graph Neural Network
GPU	Graphics Processing Unit
HMC	Hamiltonian Monte Carlo
KL	Kullback-Leibler
LES	Large Eddy Simulation
LSTM	Long Short-Term Memory
MAP	Maximum a Posteriori
MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
MFMC	Multi-Fidelity Monte Carlo
MLE	Maximum Likelihood Estimate
ML	Machine Learning
MSE	Mean Squared Error
NN	Neural Network
NUTS	No-U-Turn Sampler
ODE	Ordinary Differential Equation
OUU	Optimization Under Uncertainty
PDE	Partial Differential Equation
PGM	Probabilistic Graphical Model
PINN	Physics-Informed Neural Network
PI-TBNN	Physics Informed Tensor Basis Neural Network
PDF	Probability Density Function
QMC	Quasi-Monte Carlo
QoI	Quantity of Interest
RANS	Reynolds-Averaged Navier-Stokes
ROM	Reduced Order Modeling
SciML	Scientific Machine Learning
SMC	Sequential Monte Carlo
SVI	Stochastic Variational Inference
TBNN	Tensor Basis Neural Network
TPU	Tensor Processing Unit
UQ	Uncertainty Quantification
VB-EM	Variational-Bayes Expectation-Maximization
VI	Variational Inference

PART I

INTRODUCTION AND BACKGROUND

Not to be absolutely certain is, I think, one of the essential things in rationality.

Am I an Atheist or an Agnostic?
Bertrand Russell

1 Introduction

1.1 Motivation and overview

Scientists and engineers have long been fascinated by the complexity of natural and physical systems. The quest to understand and predict the behavior of these systems has been a driving force behind many scientific discoveries and technological advancements. A fundamental goal in any scientific and engineering discipline is the development of models that can accurately simulate the behavior of natural or physical systems. By observing the flow of water, Albert Einstein could formulate a simplified mental model of the river's complex dynamics. Einstein's anecdotal observations of a river's flow from a bridge illustrate the critical role of models in understanding nature. This approach parallels Leonardo da Vinci's detailed sketches of fluid eddies [1], where he captured their intricate patterns and behaviors.

These models can be understood as abstractions of the relationships between input factors or parameters (causes) and output variables (effects) of the system. The general framework of such a model is given in Figure 1.1. A key characteristic of these models is that their structure is deeply rooted in scientific understanding of the system, expressed through fundamental equations, domain-specific laws, rules or heuristics. In the rest of the thesis, these models are referred to as *physics-based* models. For instance, the model might be available in the form of governing equations given as an ordinary differential equation (ODE), partial differential equation (PDE), or a consortium of different models that are linked together. These diverse forms of scientific knowledge are essential components of models used in numerous application domains, including fluid dynamics, epidemiology, hydrology, geoscience, ecology, material science, climate science, and particle physics, to name a few. To elaborate the pipeline further, consider an example in computational fluid dynamics (CFD). The inputs could be the geometry of the domain, the boundary conditions, and the fluid properties. The model could be the Navier-Stokes equations or the Reynolds-averaged Navier-Stokes equations (RANS) [2], solved using some numerical method like the finite volume method (FVM) or the finite element method (FEM). The outputs could be the velocity, pressure, and temperature fields. In material science, the inputs could be the composition of the material, the processing conditions, and the microstructure, while the outputs could be the mechanical, thermal, and chemical properties.

The pipeline given in Figure 1.1 is essential for several downstream scientific tasks. These tasks include model discovery, inference, optimization, uncertainty quantification, and reduced order modeling (ROM), to name a few. These tasks are essential for scientific discovery, engineering design, and policy-making. More details about these scientific tasks are provided in Section 2.2.

Scientific and engineering fields, traditionally driven by advancements in domain-specific theories, are increasingly keen to leverage machine learning (ML) methods to aid the tasks discussed above and accelerate scientific breakthroughs [3–8]. Machine learning essentially involves the development of algorithms that can learn patterns and relationships between the input and

Inputs/ parameters
Model (e.g., PDEs)
Output/ Predictions

Figure 1.1: A simplified framework for modeling physical systems.

output. Most machine learning algorithms rely on very large datasets that are easy to obtain or readily available (*big data*). While this is, in general, a good strategy for applications where the aforementioned vast amount of data is available (e.g., computer vision, natural language processing) and has been shown to lead to very promising results [9–11], these "black-box" machine learning models have several limitations when applied to scientific problems. First, cutting-edge black-box ML methods, such as deep learning, require extensive supervision with large labeled datasets, which are rarely available in scientific contexts. Benchmark ML datasets like ImageNet contain millions of labeled images [11], a scale not feasible in most scientific problems. Second, real-world scientific data often span only a narrow segment of the possible data distributions. Consequently, even if a black-box ML model performs well on labeled data during cross-validation, it may struggle with out-of-sample data, leading to poor generalization. Third, black-box models rely solely on data for supervision and are not inherently linked to scientific theories. This detachment can result in solutions that contradict established scientific knowledge, rendering them scientifically invalid. Finally, due to their design, black-box models lack the capacity to uncover novel scientific insights from data. They do not contribute to the broader goal of advancing our understanding of scientific systems through the discovery of new knowledge.

Infusing domain knowledge in ML To address the limitations posed by the black-box ML methods in the scientific domain, there is a growing interest in the scientific community to integrate scientific knowledge into the ML frameworks. This shift is motivated by several benefits. Unlike traditional ML application areas, such as computer vision and natural language processing, which typically operate in a *big data* setting, scientific applications of ML usually fall under the *small data* regime, as the data is often obtained through costly numerical simulations or experiments. The data thus obtained are usually scarce, noisy or incomplete. The necessary information for model inference/learning can be supplemented by the knowledge of governing equations and underlying physical principles in the form of *inductive bias*. Incorporating physical knowledge and constraints directly can thus reduce the reliance on the training data. This approach of combining scientific knowledge and machine learning techniques is an emerging field known by several names, e.g., Scientific Machine Learning (SciML) [5, 6, 12], Physics Informed Machine Learning [13], Knowledge Guided Machine Learning [14]. In this thesis, we will use SciML to refer to the approach. As shown in Figure 1.2, the SciML approach aims to combine the strengths of data-driven (e.g., black-box ML) and physics-based approaches (e.g., FEM, finite difference method (FDM)) to develop models that are interpretable, generalizable, data parsimonious, and robust. SciML research is being carried out across a diverse range of disciplines and scientific objectives, for example in climate science [15–17], fluid dynamics [4, 18–20], biology [21] and particle physics [22], and expectations are rising for SciML to accelerate scientific discovery and address humanity's biggest challenges [3, 5–7]. The SciML strategies can be adopted based on the amount of scientific knowledge available. For example, in cases where the governing phenomenological equations are entirely known, it would be directly possible, given enough computational power, to simulate and reproduce the system entirely. However, in several cases, even though the governing equations are known, the high-level global dynamics are still not well understood. Consider the case of turbulent flow described by the Navier-Stokes equations. The Navier-Stokes equations are well known, but the turbulent flow is still not well understood and is a major unsolved problem [23]. The turbulent flow physics depicts multiscale and highly non-linear behavior. This makes it almost impossible to simulate/measure highly chaotic small scales, even though their impact on the physical phenomenon is critical. These small scales are more prevalent when the advection forces become increasingly dominant over diffusion, i.e., the Reynolds number increases (the flow becomes more chaotic). On the other hand, the mean-flow properties or the largest, coherent scales can still be simulated and observed, but we do not have a consistent theory to prove fundamental flow properties based on these, nor describe the detailed interactions among turbulent structures and phenomena they imply. This problem is colloquially called the *closure problem* [24], which is omnipresent when simulating multiscale systems. Addressing this problem in fluid turbulence is one of the major goals of this thesis. In the cases where partial information is available, or where the data is scarce, noisy, or incomplete, the physical knowledge can be used to regularize the learning process and improve the model performance. In the cases where governing equations are not known, the abundance of data can be used to infer the underlying physical laws from the data, e.g., social sciences and neuroscience.

Infusing inductive biases into machine learning methods, especially deep learning frameworks, is a challenging task. A popular approach to employing inductive bias is using physical laws as a regularization term, serving as a *soft constraint* to augment the loss function. The loss function is essentially the objective function that the ML algorithm seeks to optimize during the training efforts. A well-known example of adding inductive bias is Physics Informed Neural Networks (PINNs) [25], which penalizes deviations from governing laws as a residual term in model optimization. Another example includes approaches that learn an entire family of solutions by incorporating certain inputs of the PDE as inputs to the network, e.g., Deep Operator Networks (DeepONets) [26] and Fourier Network Operators (FNO) [27]. These approaches have been extensively utilized across scientific computation applications like fluid mechanics, material science, etc. However, these methods might allow the physics constraints to be violated to a certain extent. To address this, there are approaches that modify the design of ML models to treat these constraints as *hard constraints*. These often involve altering the neural network architecture to conform to invariances and symmetries in the predictions. For example, revisiting the fluid turbulence example, [28] proposed rotationally invariant tensor basis neural networks for estimating the Reynolds stress anisotropy tensor in turbulent flows. They ensured the neural network's predictions remained the same when the input axes of the flow were rotated by noting that the anisotropy tensor lies on a basis of isotropic tensors and using the neural network to predict the coefficients of the tensor in this basis. This approach provided more accurate flow predictions than a generic neural network architecture, leading to wide adoption [29,30]. *Hybrid approaches* are also gaining popularity [14,31]. They aim to integrate both physics-based and ML models tightly in a hybrid fashion to achieve superior predictive performance compared to models relying solely on ML or scientific methods.

Probabilistic Modeling The setting we are interested in involves complex physical systems that are partially observed and whose behavior could be hard to model or totally unknown. The inherent uncertainty associated with this setting necessitates a departure from the classical deterministic realm of modeling and scientific computation, and, consequently, our main building blocks can no longer be crisp deterministic numbers and governing laws, but instead we must operate with *probabilistic models* [32]. The adoption of the probabilistic approach also addresses a common challenge of ML application in the scientific domain. This challenge is the need to account for different types of uncertainties [33], such as *aleatoric* (irreducible, stems from the inherent system stochasticity) and *epistemic* (reducible, stems from incomplete knowl-

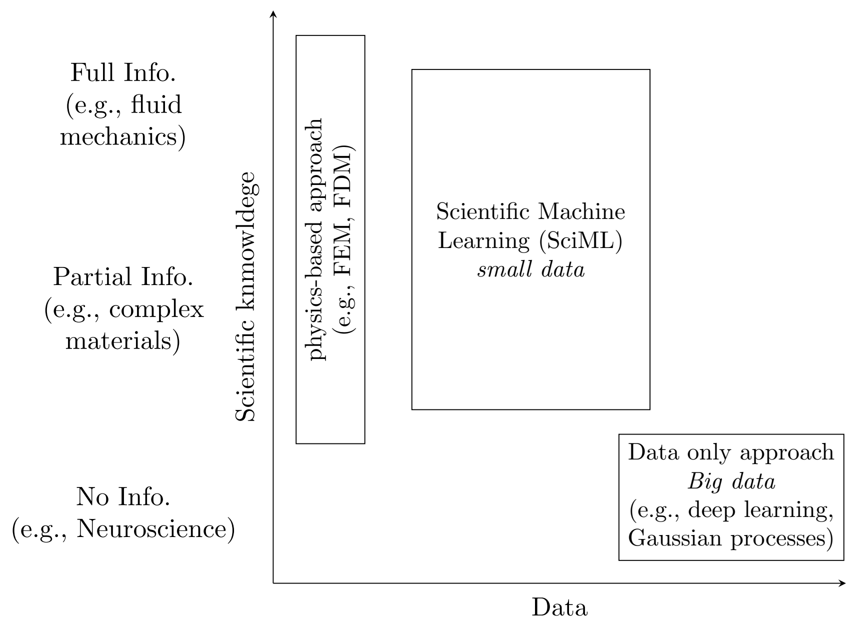


Figure 1.2: A pictorial representation of the trade-off between data-driven (black-box) and physics-based approaches. The scientific machine learning (SciML) approach aims to combine the strengths of both approaches to develop models that are interpretable, generalizable, data parsimonious and robust. Note: The area of each block carries no quantitative meaning.

edge) uncertainties, as the scientific domains typically demand high accuracy and reliability in predictions. This amalgamation of the probabilistic approach and SciML can serve as a powerful tool to address several scientific tasks like inference, model discovery etc., across scientific and engineering disciplines [8,34,35] (summarily illustrated in Figure 1.3), which we address in this work.

Differentiable Physics When tasks involving learning, inference, or optimization require the use of physics-based models, obtaining the gradient of the loss function with respect to the model parameters is a significant challenge [36]. The gradients are essential in most cases because real-world physics-based models typically have high-dimensional parameter spaces. In high dimensions, gradient-based methods are generally preferred over gradient-free methods [37]. Furthermore, when uncertainty is involved in inference or optimization, the physics-based models need to be queried an even greater number of times (e.g., sampling based expectation computation), thereby increasing the necessity for gradients. The challenges are prevalent in existing complex legacy codes that often mature over years and do not incorporate the aforementioned gradients. Recently, there has been a surge in efforts to make physics-based models differentiable, a concept known as *differentiable physics* [36,38–40]. This is a key component in the hybrid approaches of scientific machine learning (SciML) and has found applications across engineering and computational physics [22,31,39,41–44]. Differentiability is most commonly achieved through methods such as: a) the adjoint method [45], b) rewriting the physics-based model in a differentiable programming language like PyTorch, JAX, or TensorFlow to obtain gradients via automatic differentiation [46], c) training a differentiable surrogate model, or d) finite differences. Each of these methods has its benefits and shortcomings, which will be explored in detail in this thesis.

Challenges addressed This thesis aims to combine the strengths of Scientific Machine Learning, probabilistic modeling, and differentiable physics to tackle challenging applications in computational physics and engineering, as illustrated in Figure 1.3. Our research is driven by the following challenges and questions:

(i) Closure modeling: Closure problems [24] arise when a physical model that describes certain quantities of interest is created, which depends on quantities that are not of prime interest but whose effect cannot be neglected. In this work, we are particularly interested in closure problem in turbulent fluid flow. In particular, we want to answer: What kind of data should be used for learning, *directly* or *indirectly* (e.g., observations are implicitly dependent on a forward solver) observed? What to do in the limit of small data? How to parametrize the closure model? Can we use some prior physical knowledge to aid the learning process? How to best quantify the inaccuracies of the model?

(ii) Physics informed learning: How can we efficiently combine the paradigms of Machine Learning and Physics-based models to develop hybrid models that can leverage the strengths of both approaches? In the limit of small data, how can we best use the available scientific knowledge to regularize the learning process, make it data parsimonious and improve the model performance?

(iii) Probabilistic modeling: How can we account for the noise in data or model inaccuracies in the learning process in an efficient fashion, and how can we quantify the uncertainties in the predictions? As and when more data becomes available, how can we leverage the data in a Bayesian setting to improve the reliability and accuracy of the predictions? Inference in a probabilistic setting can be crippled by computational challenges, especially when physics-based models are involved. What can we do in these scenarios? Do we need a differentiable solver or do we need to make a smart selection of inference algorithms?

(iv) Differentiable physics: How can we incorporate physics-based models in ML models in a differentiable manner, so that the models can be trained end-to-end using gradient-based optimization methods? How can we optimize high-dimensional, complex, stochastic and expensive to evaluate systems under uncertainty, where the objective function is a black-box function? Can we bypass this need for differentiability?

(v) Scaling to real-world problems: What are the challenges associated with scaling learning/inference/optimization strategies to complex real-world cases, and how best to address them? Can we find some lower-dimensional latent embeddings?

1.2 Contributions

This section presents the author's contributions aimed at addressing the challenges posed in the previous section by proposing novel strategies for uncertainty quantification, scientific machine learning, and optimization for computational physics and engineering applications. The contributions are presented in the form of the author's publications, and a broad theoretical background with an up-to-date literature survey of the dynamic field of SciML and probabilistic modeling presented in Chapter 2.

The following publications have been published (or submitted/accepted for publication) prior to the submission of this thesis:

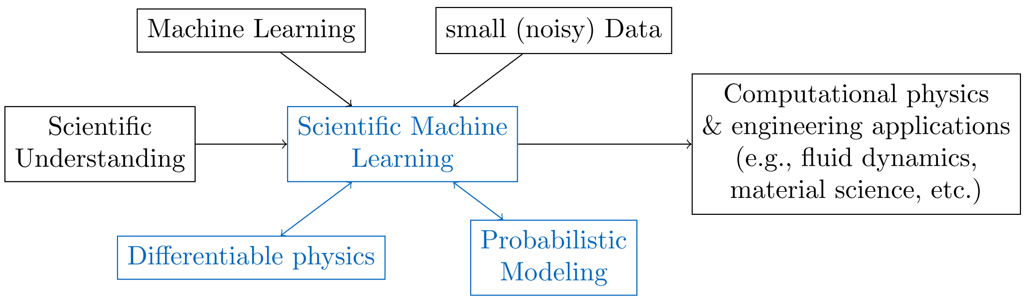

Machine Learning
small (noisy) Data
Scientific Understanding
Scientific Machine Learning
Computational physics & engineering applications (e.g., fluid dynamics, material science, etc.)
Differentiable physics
Probabilistic Modeling

Figure 1.3: A very high-level graphical overview of the aim of this thesis. The goal is to propose novel strategies that combine machine learning with scientific understanding (e.g., governing equations, symmetries, invariances etc.) to develop scientific machine learning models aided by differentiable physics, which are then tightly coupled with probabilistic approaches (combination of strategies given in blue). These strategies are then used in addressing challenging computational physics and engineering applications.

(i) Atul Agrawal and Phaedon-Stelios Koutsourelakis. A probabilistic, data-driven closure model for RANS simulations with aleatoric, model uncertainty. *Journal of Computational Physics*, page 112982, 2024 (Paper A [47])

(ii) Leon Riccius, Atul Agrawal, and Phaedon-Stelios Koutsourelakis. Physics-informed tensor basis neural network for turbulence closure modeling. *Workshop on Machine Learning and the Physical Sciences (NeurIPS 2023)*, 2023 (Paper B [48])

(iii) Atul Agrawal, Erik Tamsen, Phaedon-Stelios Koutsourelakis, and Joerg F Unger. From concrete mixture to structural design–a holistic optimization procedure in the presence of uncertainties. *Data-Centric Engineering*, pages 1–32, 2024 (Paper C [49])

(iv) Atul Agrawal, Kislaya Ravi, Phaedon-Stelios Koutsourelakis, and Hans-Joachim Bungartz. Multi-fidelity constrained optimization for stochastic black-box simulators. *Workshop on Machine Learning and the Physical Sciences (NeurIPS 2023)*, 2023 (Paper D [50])

(v) Atul Agrawal, Kislaya Ravi, Phaedon-Stelios Koutsourelakis, and Hans-Joachim Bungartz. Stochastic black-box optimization using multi-fidelity score function estimator. *Machine Learning: Science and Technology*, 6(1):015024, jan 2025 (Paper E [51])

To position our contributions in the broad skeleton of SciML, the publications are categorized based on the SciML problems, SciML methodologies employed and application domain of focus. This categorization is presented in the table 1.1. To this end and in light of the categorization, below we provide a brief summary of the major accomplishments in this cumulative thesis.

We approach the *turbulence closure problem* in Paper A [47] and Paper B [48]. We realized addressing the closure problem required overlap of all the three broad strategies i.e., SciML, probabilistic modeling, and differentiable physics (as discussed in Figure 1.3). Closure modeling requires mapping the effect of small scales on the large scales which involves non-trivial nonlinear approximations, for which the hybrid approach of SciML serves to be a very promising candidate [52]. Also, in the context of closure modeling, latent(unobserved) variables are to be inferred and the model is learned using potentially limited/sparse, noisy or indirectly observed data, suggesting a probabilistic approach. Furthermore, owing to the high dimensional parametric space in multiscale problems, a gradient-based learning scheme is often desired, suggesting the need for differentiable physics. In particular, Paper A [47] attempts to touch on all the challenges posed in Section 1.1 in the context of turbulence closure modeling, and has the following main contributions:

Chapter	**SciML problem (*What?*)**	**Methodology (*How?*)**	**Application**
Paper A [47]	Closure modeling (model discovery), Uncertainty Quantification, Inverse problem	Hybrid approach, Physics guided ML architecture, differentiable physics	Turbulence closure problem
Paper B [48]	Closure modeling (model discovery), Inverse problem	Physics guided ML architecture	Turbulence closure problem
Paper C [49]	Uncertainty Quantification, Inverse problem, model discovery, Optimization	Hybrid approach	Design optimization of concrete mixture and concrete structure
Paper D [50]	Optimization	Differentiable physics	Academic examples
Paper E [51]	Optimization	Differentiable physics	Academic examples, windfarm layout optimization

Table 1.1: Categorization of the contributions by the author in this work based on the SciML problems, methodologies employed, and application domain of focus. Further details and other related works on the SciML problems, and methodologies are given in Section 2.2 and the respective publication by the author.

- We propose a probabilistic, data-driven *closure model* for Reynolds-Averaged Navier-Stokes simulations.
- Learning is performed in a Bayesian setting, with *indirect* high-fidelity data of mean velocity/pressure as the training data.
- It follows the hybrid approach of SciML, i.e., it combines the RANS solver and ML model in a tightly coupled fashion. To enable end-to-end gradient-based learning, it makes use of an adjoint-based differentiable RANS solver.
- The ML model is designed in such a way that physical constraints, such as symmetry and invariance, are treated as hard constraints.
- It infers the statistics of a *reduced latent variable* that can automatically identify regions where closure is incorrect and provides stochastic corrections.
- The training is data parsimonious and the probabilistic predictive estimates envelop the reference values obtained from higher-fidelity simulations.

Paper B [48] address the challenges (i) and (ii) (as given in Section 1.1) and has the following main contributions:

- This work introduces Physics Informed Tensor Basis Neural Network (PI-TBNN), which extends the TBNN framework with an extensive feature set and an inductive bias in the form of a physics-informed addition to the loss function (soft constraint) to improve anisotropy tensor predictions.
- Learning is performed with sparse observations of the Reynolds stress tensor.

- The ML model is designed in such a way that physical constraints, such as symmetry and invariance, are treated as hard constraints.
- The proposed machine learning method significantly enhances anisotropy tensor predictions in unseen challenging cases involving surface curvature and flow separation.

In Paper C [49] we present a methodological paper that attempts to synthesize physics-based models, empirical relations and experimental data which are generally employed in a disjointed manner in the design of structural systems made of concrete. The paper attempts to address the challenges (ii), (iii), (iv) and (v) (as given Section 1.1). It has the following main contributions:

- The introduction of a systematic design framework for precast concrete that promotes sustainability.
- The development of a holistic optimization framework that combines mixture design and structural simulation, consisting of semi-analytical models, finite-element solvers and machine learning models.
- A model discovery method to learn missing links between concrete mixture design variables and parameters appearing in physics-based models by making use of noisy and incomplete, experimental data and finite element simulations.
- An optimization framework that can handle black-box solvers, non-linear constraints as well as parametric uncertainties in the workflow.
- Application in the mixture design of a concrete beam with the objective being the Global Warming Potential while satisfying a variety of performance constraints.

Paper D [50] and Paper E [51] propose efficient strategies to perform constraint optimization when a stochastic black-box physics-based model is involved. The Paper D [50] presents a precursory work, which is then expanded in Paper E [51]. The Paper D [50] addresses challenge (iv) (as given in Section 1.1) and has the following main contributions:

- We propose an algorithm for stochastic constraint optimization with objectives and constraints involving black-box physics-based models.
- The algorithm relies on efficient gradient estimation with variance reduction strategies.
- To alleviate the problem of multiple evaluations of the expensive physics-based models thus leading to a trade-off between computational cost and accuracy, multi-fidelity strategies are proposed.
- The proposed method performs better than standard black-box optimization methods on average, demonstrating improved robustness and quality of the optimum when tested on academic examples.

Paper E [51] inherits the contributions from the Paper D [50], addresses challenges (iv) and (v) and extends it by providing the following main contributions:

- Achieves well-behaved convergence properties and better quality of the optimum by using natural gradients in the gradient estimation process.
- Utilizes smartly chosen heuristics to efficiently handle multi-modalities, as demonstrated using academic examples.

- Provides a detailed mathematical analysis of method convergence.
- Implements an adaptive sample size for gradient estimation, reducing the number of solver calls needed to arrive at the optimum in numerical tests.
- Includes an expanded academic example set to test the method against standard baselines for the algorithm's ability to handle different types of optimization challenges like multi-modality, constraints, behavior in valleys, and dimension scalability.
- Demonstrates improved performance in a real-world complex case of wind farm layout optimization, outperforming the standard baseline method in terms of robustness and the quality of the optimum.

A summary of the publications is given in Chapter 3 along with the author's individual contributions. The full text of the publications is attached in the respective appendix chapters in Part III.

1.3 Scope of this work

This thesis encapsulates aspects from applied mathematics, uncertainty quantification, machine learning, computational physics, scientific computing and optimization. We tried to maintain a balance between theoretical rigor and ease of explanation, and breadth and depth of the topics being discussed. In terms of putting concepts in perspective, we provided a wide overview of the concepts being discussed, while directing the reader to the relevant literature for further reading. For reasons of brevity, this thesis does not give a detailed introduction to the scientific and engineering applications being addressed. For instance, the turbulence phenomenon, fluid dynamics and the numerical strategies to solve the corresponding governing equations like the finite volume method, finite element method, etc. are not discussed in detail. All of this is well-documented in the literature. Similarly, computational challenges involved in concrete technology are also well studied. Our aim is mostly to discuss the novel strategies proposed in this work, how they address the challenges posed when physics-based models are coupled with machine learning frameworks, and the promise they bring in the context of the broader scientific machine learning, uncertainty quantification and optimization literature.

1.4 Thesis outline

The remainder of this thesis is structured as follows. Chapter 2 provides the necessary background relevant to this thesis. In particular, we commence with probabilistic modeling and uncertainty quantification, covering broad areas such as uncertainty propagation and Bayesian inference. Then, we discuss Scientific Machine Learning (SciML), the problems addressed by SciML, major categorizations of SciML, and related topics like differentiable physics and closure modeling. In Chapter 3, we summarize the publications presented as part of this thesis along with the author's contributions. The thesis concludes in Chapter 4, where we summarize the findings from the aforementioned publications and provide an outlook for future research. The appendix chapters A, B, C, D and E in part III provides the full text of the publications.

If you wish to make an apple pie from scratch, you must first invent the universe.

Carl Sagan

2 Background

In this chapter, we overview the fundamental concepts and methodologies used in this work and provide relevant literature suggestions for further reading. We start with a brief introduction to the Bayesian probability theory and Uncertainty Quantification in Section 2.1. The section motivates the need for probabilistic modeling, discusses the types of uncertainties, and briefly discusses uncertainty propagation and Bayesian inference. In Bayesian inference, we discuss the prior, likelihood, and posterior, and the role of the Bayes theorem in updating the prior knowledge with the observed data. We also discuss the Markov Chain Monte Carlo (MCMC) methods and the variational inference methods used in Bayesian inference. Then in Section 2.2, we discuss Scientific Machine Learning (SciML) and the promise it offers. More specifically, we discuss the major scientific tasks addressed by SciML like inverse problems, model discovery, surrogate modeling, reduced order modeling, optimization etc. We provide a brief overview of Machine Learning (ML) and methods being developed in SciML like the Physics-Informed Neural Networks (PINNs), Deep Operator Networks (DeepONets), hybrid models etc. We also discuss closely related topics like differentiable physics and closure modeling. We remark that in the literature on uncertainty quantification, machine learning, and physics-informed machine learning different notations are commonly used. In this work, we use a unified notation that should be clear from the context, while trying to keep standard notations whenever possible.

2.1 Bayesian probability theory and Uncertainty Quantification

Probabilistic reasoning is essential not only in scenarios where stochastic phenomena are being studied but also in any context where reasoning must be conducted with finite and incomplete information. This need arises because, in many real-world situations, we lack full knowledge or face inherent randomness, requiring us to make informed decisions or predictions based on available data. As quoted by [32], probabilistic methods are the "logic of science". It provides a structured way to quantify uncertainty, enabling more reliable and nuanced decision-making across various fields, from science and engineering to economics and social sciences. When probability theory is used in the context of plausible reasoning (or degree of belief), it is referred to as *Bayesian* probability theory [53]. The Bayesian probability theory is a mathematical framework for representing uncertain beliefs and updating these beliefs in the light of new evidence. Another comprehensive term employed in this thesis is the *Uncertainty Quantification* (UQ), which roughly put by [54] is "the coming together of probability theory and statistical practice with 'the real world'." To put it differently, "A more precise definition of UQ is the end-to-end study of the reliability of scientific inferences" [55]. UQ is in general a vast field with ever changing landscape, more so nowadays with the advent of ML and AI. Quoting Sullivan [54], "UQ is much more about problems, methods and 'good enough for the job'. There are some very elegant approaches within UQ, but as yet no single, general, over-arching theory of UQ." In light of the insight, the present section provides a brief overview of the methods relevant only to this thesis. We will limit ourselves to two broad objectives. First, the forward propagation of uncertainty (discussed in Section 2.1.3). Second, the inverse problem of estimating the input parameters θ given the corrupted or noisy output $\hat{\boldsymbol{y}}$ (discussed in Section 2.1.4). These two broad objectives overlap and cover other problems also. Consider for example, one might have to solve an inverse problem to produce or improve some model parameters, and then use those parameters to propagate some other uncertainties forward, and hence produce a prediction that can be used for decision support in some certification problem. We do not attempt to discuss basic concepts of probability theory and defer the reader to [32, 54, 56–58] instead. For probabilistic machine learning, the reader is referred to [59–61].

2.1.1 Notational conventions

In this work, we deal with complex, non-linear and multiscale systems like turbulent flows for example. These problems are typically governed by a model f representing e.g., a complex and nonlinear ODE/PDE system derived from first principles. The analytical solution to this model is very rarely available, hence numerical approximation is performed by employing discretization strategies, which is then solved numerically. Thus such models *implicitly* enter the inference/optimization. Let us say the discretization operators describing the model f are given by f_h, for the sake of brevity we drop the subscript in the subsequent discussions. The model f depends on fixed *deterministic inputs* $\boldsymbol{x}$ like time, spatial coordinates, initial, boundary condition etc., and *stochastic inputs* $\boldsymbol{\theta}$. Since $\boldsymbol{x}$ is fixed, we drop this notation and denote the model as $f \equiv f(\boldsymbol{\theta})$, with the outputs given by $\boldsymbol{y}$ in space $\mathcal{Y}$, so we have $\boldsymbol{y} = f(\boldsymbol{\theta})$. We further assume that the model f is a bounded and measurable function w.r.t. the Lebesgue measure. The input random vector $\boldsymbol{\theta}$ is continuous, characterized by a *propability density function* (PDF) $p(\boldsymbol{\theta})$ defined in an appropriate probability space with support $\boldsymbol{\Theta}$, i.e., $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. Also $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{d_\theta}) \subset \mathbb{R}^{d_\theta}$, where d_θ is referred to as the *stochastic dimention.* Additionally, the components of $\boldsymbol{\theta}$ are assumed to have finite expectations and variances.

2.1.2 Sources and Types of Uncertanity

Determining how much uncertainty there is in a model's output, and identifying where this uncertainty comes from, is crucial in many areas of science and engineering, particularly when it comes to making decisions in the real world. For instance, in aircraft design, understanding the level of uncertainty in aerodynamic performance predictions under varying input scenarios (e.g., wind conditions), helps engineers mitigate risks and design more efficient, safer aircraft. The discipline of uncertainty quantification has evolved over many years, encompassing a wealth of research from numerous scientific fields. This body of work spans statistical and machine learning methodologies, including Bayesian modeling and variational inference. We typically divide sources of predictive uncertainty into two groups, the first is *aleatoric* (or irreducible), which is the inherent uncertainty of the system. It says the model's behavior is truly random and is characterized by the fact that no more information about the model and its parameters can reduce this uncertainty. Examples of such uncertainty are the chaotic behavior of some nonlinear dynamic systems and the turbulent behavior of fluids. The second is *epistemic* (or reducible), which stems from incomplete knowledge (discussed more in the paragraph below). While both categories are integral to the study of scientific systems, the intricacies of uncertainty quantification in Scientific Machine Learning (SciML) are particularly nuanced. In SciML, the challenge is heightened as uncertainties can emerge and multiply due to the combination of data and scientific theory, potentially leading to compounded errors. Next, we will delve into the two primary sources of uncertainty within SciML contexts.

Model Uncertanity In engineering and physical systems, one typically has a model to describe some system of interest, then epistemic uncertainty is often further subdivided into *model form* and *parametric* uncertainty. In model form uncertainty, one has significant doubts that the model is even *structurally correct*. In parametric uncertainty, one believes that the form of the model reflects reality well, but one is uncertain about the correct values to use for particular parameters in the model. For instance, both these types of uncertainties are very common whenever a closure model is used in a simulation [24,52]. In this case, the closure model is often a simplified version of the true physics (model form uncertainty), and the parameters in the closure model are often not known exactly (parametric uncertainty). This is a very active line of research in the field of turbulence closure modeling [4, 20, 24], where we also contribute (Paper A [47]).

Model form and parametric uncertainty can also appear in ML models. For example, given a finite set of data samples, several choices of ML hypotheses (model structures and parameters) can fit the data, resulting in model form uncertainty. However, not every parameter selection in these ML models may be capable of producing predictions that align with scientific knowledge, even if they perfectly match the available data. Excluding such scientifically incompatible solutions from the pool of potential hypotheses could potentially aid in diminishing model uncertainty. Therefore, in SciML problems, scientific knowledge can serve as an additional supervisory mechanism to decrease the uncertainty in model predictions caused by the choice of ML model parameters that contradict established knowledge.

Data Uncertainty This deals with the *measurement noise* due to the errors in sensor precision (*aleatoric uncertainty*). This is intrinsic to the process of collecting observations. For instance, in the context of fluid dynamics, the velocity measurements obtained from a Particle Image Velocimetry (PIV) system are subject to measurement noise. This noise can be due to the resolution of the camera, the quality of the laser sheet, or the seeding particles. In addition to measurement noise, data uncertainty can also arise from the *sparsity* of the data. In many scientific applications, the data is often sparse, and the model must make predictions in regions where no data is available. This is a common challenge in fluid dynamics, where the data is often expensive to collect and is limited to a few locations. In such cases, the model must make predictions in regions where no data is available.

2.1.3 Uncertainty Propagation: an overview

Following the notational convention introduced in Section 2.1.1, assuming f is well posed[1], the task of uncertainty propagation amounts to quantifying the effect of input $\boldsymbol{\theta}$ uncertainty on the output $\boldsymbol{y}$ of the forward model, i.e., determining the induced probability distribution $p(\boldsymbol{y})$ on the output space $\mathcal{Y}$. Uncertainty is thus propagated through f in the direction input $\rightarrow$ model $\rightarrow$ output. This task is typically complicated by $p(\boldsymbol{\theta})$ being a complicated distribution (e.g., multimodel distribution, heavy tail), or f being non-linear, expensive to evaluate, and implicitly defined (e.g., solving a PDE to get the solution field). Because $p(\boldsymbol{y})$ can be a very high-dimensional object, it is often more practical to identify some specific outputs of interest obtained by the action of a functional on $\boldsymbol{y}$. For brevity, we refrain from introducing a new notation and assume that from the context it should be clear that $\boldsymbol{y}$ can represent both the output of a function f and the output of a functional (then f would be a composite function). For example, in a fluid flow problem, the input can be a stochastic boundary condition, the model can be a fluid flow solver, the outputs can be pressure and fluid velocity and the output

[1]The forward mapping is assumed to be well-posed i.e., a solution exists, it is unique and the solution depends continuously on the data.

of interest can be the drag force on the body, i.e., the integrated surface pressure in direction of the fluid flow.

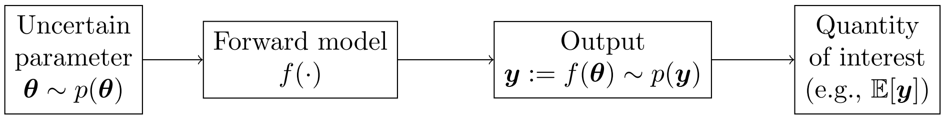

Uncertain parameter $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$
Forward model $f(\cdot)$
Output $\boldsymbol{y} := f(\boldsymbol{\theta}) \sim p(\boldsymbol{y})$
Quantity of interest (e.g., $\mathbb{E}[\boldsymbol{y}]$)

Figure 2.1: *Forward uncertainty propagation workflow.* We sample input from its prior distribution $p(\boldsymbol{\theta})$, solve the forward model f (can be an ODE, PDE solved with Finite element methods for example) to obtain the output of interest $\boldsymbol{y}$ for each sample. The propagation of the sample from the input distribution through the model to the output of interests $\boldsymbol{y}$ gives rise to the probability distribution of $\boldsymbol{y}$ given by $p(\boldsymbol{y})$. Finally, the quantities of interest are computed from the ensemble of the forward model calls.

For the forward uncertainty propagation, it is common to compute *robustness measures* [62] such as expectation $\mathbb{E}[\cdot]$ and variance $\mathrm{Var}[\cdot]$ or standard deviation $\mathrm{Std}[\cdot]$. given by:

$$\mathbb{E}_{p(\boldsymbol{\theta})}[\boldsymbol{y}] := \int_{\Theta} \boldsymbol{y}(\boldsymbol{\theta}) p(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}, \tag{2.1}$$

$$\mathrm{Var}[\boldsymbol{y}] := \mathbb{E}[\boldsymbol{y}^2] - (\mathbb{E}[\boldsymbol{y}])^2, \quad \mathrm{Std}[\boldsymbol{y}] := \sqrt{\mathrm{Var}[\boldsymbol{y}]}. \tag{2.2}$$

Such quantities are often called *quantities of interest* (QoI). For example, QoI can include expected rainfall amounts for a specified area, expected temperature increase over a future time period, or bounds on void fraction distributions that guarantee specified performance levels and safety margins in a nuclear reactor. The workflow of the forward uncertainty propagation is shown in Figure 2.1. Note that in this thesis we might drop the subscript from $\mathbb{E}_{p(\boldsymbol{\theta})}[\cdot]$ for brevity, without the loss of meaning.

Commonly, the uncertainty propagation techniques are broadly the following categories. a) Direct evaluation for linearly parametrized models, for example in image processing and X-ray tomography. The QoIs like expectation and variance can be computed explicitly in closed form in this case. b) Sampling methods. These methods, including Monte Carlo techniques, are commonly employed to propagate uncertainties in nonlinear problems. (discussed more in next to subsequent paragraphs). c) Perturbation method [54], Taylor expansions of the model response or QoI, truncated to a certain degree, are evaluated at the mean of the parameters. To simplify the implementation, expansions are usually limited to the first or second order. However, this can restrict the accuracy of the technique in scenarios where the relationship between inputs and responses is highly nonlinear. and d) Spectral representations, the uncertain input is represented in a manner that facilitates the evaluation of moments and distributions for QoI, usually by stochastic Galerkin and collocation methods [63]. Spectral expansions are employed to exploit the smoothness associated with the high-dimensional parametric spaces, to improve the convergence of techniques used to specify realizations used to specify QoI (e.g., Karhunen-Loéve expansions [63], Polynomial Chaos [64]). For more details, interested readers are referred to Chapter 9, 10 [56] and Chapter 12, 13 [54].

Intrusive and Non-Intrusive Based on the type of physical model involved, the uncertainty propagation methodologies can also be categorized as *intrusive* and *non-intrusive* [54]. Intrusive techniques necessitate modifications to the model's core operators, subsequently requiring alterations to the simulation code itself. Many times the solution to simulations is slow and expensive to obtain, and the deterministic solution method cannot be modified. Many complex simulation codes or so-called *legacy codes* are not amenable to such intrusive methods of UQ. Consequently, intrusive UQ methods are not considered within our scope. Conversely, non-intrusive approaches, such as Monte Carlo (MC) sampling or collocation-based strategies, rely on sets of independent, high-fidelity evaluations that treat the underlying models as black boxes. Our investigation is strictly focused on non-intrusive algorithms for propagating uncertainty. Henceforth, the term ' forward uncertainty propagation' in this document will exclusively denote non-intrusive methods.

Monte Carlo and Quasi Monte Carlo methods This thesis mostly deals with non-linear problems with correlated parameters or sufficiently large parameter dimensions, which makes the Monte Carlo (MC) methods a natural choice for uncertainty propagation [56,65]. Essentially the MC method provides approximation to the integral given by Eq. (2.1) in the form of sample-based integration. Its accuracy does not depend on the dimension of the integration domain ($dim(\boldsymbol{\theta})$), but rather on the variance of the output and the number of samples. The other two forms of the integral approximation are classical grid-based quadrature in which nodes of the parameter space are determined in a deterministic fashion (e.g., univariate quadrature, Gaussian quadrature etc.) and quasi-Monte Carlo (QMC) methods [66] (discussed later in more detail). The grid-based quadrature suffers from the curse of dimensionality, i.e., they require exponentially many evaluations of the integrand as a function of the dimension of the integration domain. Remarkably, however, the curse of dimensionality can be entirely circumvented by resorting to sampling based strategies like MC or QMC method — provided, of course, that it is possible to draw samples from the measure w.r.t. which the integrand is to be integrated.

The MC method works by drawing samples from the input parameter distribution and evaluating the model at each sample. The distribution of the output is then estimated by computing the empirical distribution of the model evaluations. To approximate the integral given in Eq. (2.1) by the *vanilla* Monte Carlo method, the algorithm is given as:

- sample $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta})$ for $i = 1, 2, \ldots, S$,
- unbiased estimate of the integral is given by $I \approx I_S = \frac{1}{S}\sum_{i=1}^{S} \boldsymbol{y}(\boldsymbol{\theta}^{(i)})$,

which converges by the law of large numbers for $S \to \infty$ to $I_S \to I$, where S is the number of samples drawn from the input distribution. The sampling based methods are intuitive, easy to implement and they are non-intrusive. As an additional advantage, their efficiency is independent of the number of parameters since one can simultaneously sample from each parameter distribution. The disadvantage of sampling methods is that they typically exhibit relatively slow convergence rates, and thus a large number of physical model realizations are required to construct a reasonable statistical ensemble. The challenge is exacerbated when the model is computationally expensive to evaluate as in engineering and physics applications dealt with in the present thesis. For example, MC methods have an $\mathcal{O}(S^{-1/2})$ convergence rate. In contrast, QMC methods have a convergence rate close to $\mathcal{O}(S^{-1})$. To be precise, the convergence rate of the QMC is $\mathcal{O}\left(\frac{(\log S)^{d_\theta}}{S}\right)$. This elucidates that for the convergence rate of the QMC to be smaller than the MC, the parametric dimension d_θ needs to be small and the number of samples S needs to be large. The QMC methods are 'approximately random' in the sense that they are deterministic, but they are designed to mimic the properties of random numbers. The QMC methods are based on the idea of low-discrepancy sequences [54], which are deterministic sequences that have better coverage properties than random sequences. In practice, it is almost always possible to select an appropriate low-discrepancy sequence, to ensure that QMC performs at least as well as MC (and often much better) [67]. Another class of algorithms that improves upon the vanilla

MC estimator is the *Multi-Fidelity Monte Carlo* (MFMC) [68]. MFMC exploits the fact that in most problems we have available surrogate or low-fidelity models which approximate the given high-fidelity model, thus reducing the variance of standard MC sampling via an expression that depends only on the correlation coefficients and computational costs of the underlying hierarchy of models. For a detailed treatment of MFMC, the reader is referred to [68].

In summary, we motivated forward uncertainty propagation and discussed methods in brief. For an extended and more in-depth discussion, the interested reader is referred to [54, 56, 63, 66, 69]. In the next section, we will discuss the Bayesian inverse problem of estimating the input parameters given the output data.

2.1.4 Bayesian Inference

This section provides an introduction at a high level to the backward propagation of uncertainty in the solution of *inverse problem*. More specifically, we will provide a Bayesian probabilistic perspective to the inverse problems involving parameter estimation and regression.

Following the notational convention introduced in Section 2.1.1, the task of Bayesian inference is to estimate the input parameters $\boldsymbol{\theta}$ of the model f for fixed inputs given some observations of the output $\hat{\boldsymbol{y}}$ which may be corrupted or unreliable in some way. In the Bayesian Inference context, let $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^N$ be a separable Banach space that represents the space of observations. Since the parameter and data spaces are finite-dimensional, this configuration enables the use of densities with respect to the Lebesgue measure. Very often we do not actually observe $f(\boldsymbol{\theta})$, but some noisy corrupted version of it, given by $\hat{\boldsymbol{y}} \in \boldsymbol{\mathcal{Y}}$. The noise is assumed to be additive and characterized by a non-degenerate Gaussian distribution with mean zero and covariance matrix $\boldsymbol{\Sigma}$, with $\boldsymbol{\Sigma}$ being symmetric and positive definite. The task for the inverse problem thus amounts to identifying the parameter $\boldsymbol{\theta}$ for the equation

$$\hat{\boldsymbol{y}} = f(\boldsymbol{\theta}) + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(0, \boldsymbol{\Sigma}). \tag{2.3}$$

The above is typically *ill-posed*[2] [70], due to, a) the presence of noise in the data with adversely affects the solution stability, b) the availability of only *small* data which is often not enough to allow the identification of a unique parameter in high-dimensional space $\boldsymbol{\Theta}$, and c) non-convex function f can have an infinite amount of inverse mapping solutions. All these scenarios are prevalent in scientific and engineering applications. The ill-posedness can be cured by *regularization*, such as *Tikhonov regularization* [71]. However, the regularization parameter is often difficult to choose, and the regularization can introduce bias in the solution [59]. Furthermore, this approach does not provide a measure of uncertainty in the estimated parameters. Taking a probabilistic viewpoint alleviates these difficulties. Jayes aptly observed “probability distributions are carriers of incomplete information” [32]. In this thesis, we adopt a Bayesian approach to address the ill-posedness of the inverse problem.

Bayes’ Theorem If probability is a measure of the degree of belief about a proposition, then Bayes’ theorem is a rule for updating that belief in the light of new evidence. The theorem is named after Thomas Bayes, who first formulated it in the 18th century. Although Bayes’ theorem is a straightforward consequence of the definitions of conditional probabilities or densities, its significance cannot be overstated. This theorem is fundamental to the process of inference that underpins probabilistic reasoning and machine learning. Therefore, we will explore several implications and properties that are implicitly contained within Bayes’ theorem.

[2]An inverse problem is deemed ill-posed if it does not satisfy one or more of the following criteria: a) Existence: A solution exists. b) Uniqueness: The solution is unique. c) Stability: The solution changes continuously with the input data.

Let us consider the parameters $\boldsymbol{\theta}$ being distributed according to a *prior* distribution $p(\boldsymbol{\theta})$. We assume $\boldsymbol{\theta}$ is stochastically independent of the noise $\boldsymbol{\eta}$. The forward model $f(\cdot)$ provides possible explanation of the data $\mathcal{D} = \{\hat{\boldsymbol{y}}_i\}_{i=1}^N$, that is embedded in a *likelihood* function $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^N \mathcal{N}(\hat{\boldsymbol{y}}_i|f(\boldsymbol{\theta}), \boldsymbol{\Sigma})$ (since we assume a Guassian additive noise). We intend to make statements about the plausibility of parameters (and/or competing models/hypotheses) given the observation of a dataset $\mathcal{D}$. The *posterior* distribution of the parameters can now be compounded using the Bayes's formula:

$$\underbrace{p(\boldsymbol{\theta}|\mathcal{D})}_{posterior} = \frac{\overbrace{p(\mathcal{D}|\boldsymbol{\theta})}^{likelihood}\ \overbrace{p(\boldsymbol{\theta})}^{prior}}{\underbrace{p(\mathcal{D})}_{evidence}}, \quad \boldsymbol{\theta} \in \boldsymbol{\Theta}, \quad \mathcal{D} \in \boldsymbol{\mathcal{Y}}. \tag{2.4}$$
$$\text{where, } p(\mathcal{D}) = \int_{\boldsymbol{\Theta}} p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta},$$

provided $0 < p(\mathcal{D}) < \infty$. In the given setting, i.e., non-degenerate Gaussian additive noise, finite-dimensional data space, one can show that the evidence is always finite and bounded away from 0. This implies the existence of the posterior measures [72]. Following our discussion of the Bayesian approach remedying the ill-posedness of the inverse problem, [72] also establishes that Bayesian inverse problems of this type are always *well-posed* (also see Chapter 6 [54]). Bayes theorem is in contrast to the *maximum likelihood* approach which seeks to identify the parameters $\boldsymbol{\theta}^*$ that satisfy $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})$. The maximum likelihood approach does not provide a measure of uncertainty in the estimated parameters as opposed to the Bayesian approach.

Before we move forward, we will discuss the three densities introduced in this context i.e., the prior, likelihood and the evidence. The prior density reflects our knowledge about the parameters before we make an observation. Intuitively, the prior regularizes the inverse problem, restricting the space in which the sought parameters lie. For instance, the prior can encode spatial correlation structure in parameters in problems pertaining to fluid mechanics. The likelihood function quantifies the probability (or probability density) of observing a specific dataset, given the parameters and the model/hypothesis. In conjunction with the prior, the likelihood scores plausibility of the parameters $\boldsymbol{\theta}$ by their ability to explain the observed data $\mathcal{D}$. The evidence is the normalization constant that ensures the posterior integrates to 1. It is the probability of observing the data averaged over all possible parameter values. The evidence is often intractable in practice, as it involves integrating the likelihood over the entire parameter space, thus necessitating the use of approximation methods discussed in Section 2.1.5.

After obtaining the posterior distribution (or a sufficiently accurate approximation), one may employ it to make predictions by means of *posterior predictive* distributions. The posterior predictive distribution is the distribution of a new observation $\hat{\boldsymbol{y}}'$ given the observed data $\mathcal{D}$. It is obtained by integrating the likelihood over the posterior distribution of the parameters:

$$p(\hat{\boldsymbol{y}}'|\mathcal{D}) = \int_{\boldsymbol{\Theta}} p(\hat{\boldsymbol{y}}'|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D})\, \mathrm{d}\boldsymbol{\theta}. \tag{2.5}$$

The above essentially marginalizes out the epistemic uncertainty in our model parameters. The posterior predictive $p(\hat{\boldsymbol{y}}'|\mathcal{D})$ and posterior $p(\boldsymbol{\theta}|\mathcal{D})$ capture all the relevant information available about the model and model predictions given the data $\mathcal{D}$. This information is crucial for making predictions, informing decisions, and evaluating competing designs in the engineering domain. Moreover, one may evaluate any arbitrary expectation contingent on the inferred parameter $\boldsymbol{\theta}$ now accessible by the posterior distribution and an arbitrary forward model $\tilde{f}$. Following our discussions and notations from Section 2.1.3, the quantity of interest $\mathbb{E}[\boldsymbol{y}]$ can now be computed as $\mathbb{E}[\boldsymbol{y}] = \int_{\boldsymbol{\Theta}} \boldsymbol{y}(\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D})\, \mathrm{d}\boldsymbol{\theta}$. To summarize, we provide a visual representation of the Bayesian inference workflow leading to the posterior predictive in Figure 2.1.

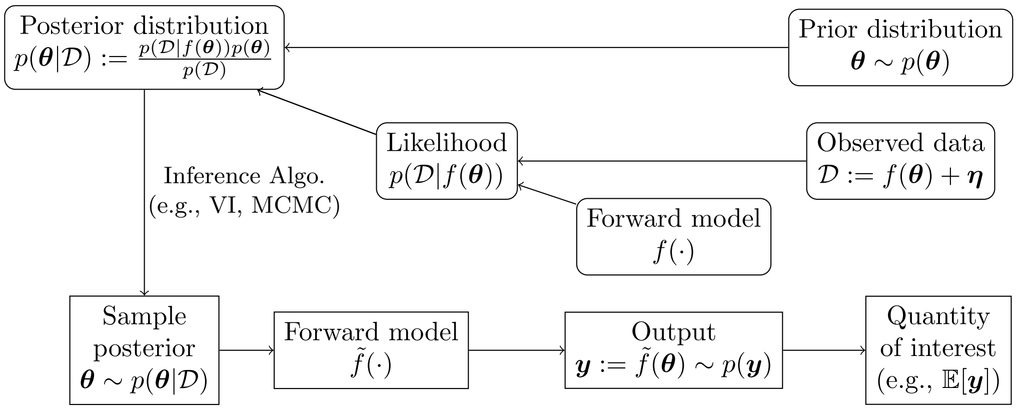

Posterior distribution $p(\boldsymbol{\theta}|\mathcal{D}) := \frac{p(\mathcal{D}|f(\boldsymbol{\theta}))p(\boldsymbol{\theta})}{p(\mathcal{D})}$
Prior distribution $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$
Likelihood $p(\mathcal{D}|f(\boldsymbol{\theta}))$
Observed data $\mathcal{D} := f(\boldsymbol{\theta}) + \boldsymbol{\eta}$
Inference Algo. (e.g., VI, MCMC)
Forward model $f(\cdot)$
Sample posterior $\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\mathcal{D})$
Forward model $\tilde{f}(\cdot)$
Output $\boldsymbol{y} := \tilde{f}(\boldsymbol{\theta}) \sim p(\boldsymbol{y})$
Quantity of interest (e.g., $\mathbb{E}[\boldsymbol{y}]$)

Figure 2.2: *Bayesian parameter inference and posterior predictive.* (rectangle with rounded edges) Given the forward model $f(\cdot)$, prior density $p(\boldsymbol{\theta})$, and the experiemntal data $\mathcal{D}$ affected by the experiemntal noise $\boldsymbol{\eta}$, the Bayesian approach finds a posterior density $p(\boldsymbol{\theta}|\mathcal{D})$ for the parameters $\boldsymbol{\theta}$. (rectangle with sharp edges) The posterior is then used to sample the parameters and evaluate any arbitrary forward model $\tilde{f}$ to obtain the quantity of interest $\mathbb{E}[\boldsymbol{y}]$.

Before we move to the next section, we will briefly introduce some concepts that are relevant to the inference algorithms.

On the choice of prior The choice of the prior is a crucial step when applying Bayes' theorem. One may opt for a broad, less informative prior or choose a more restrictive, informative prior that confines parameters to a specific range. In the context of physical systems, the latter is often preferred due to existing knowledge about parameter characteristics—for instance, certain parameters must be non-negative, or physical constraints may limit parameters to specific intervals. When such prior information is available, it is advantageous to incorporate it by selecting an appropriate prior. In scenarios where the parameter space is high-dimensional and only a few parameters are expected to deviate from zero, it is advisable to use priors that enforce sparsity. These priors typically have heavy tails, which help prevent overfitting by promoting a sparse representation of the parameters. A widely used method in this context is the Automatic Relevance Determination (ARD) [73] framework and its extensions [74–76], which effectively identifies and isolates the most relevant variables in a model. Specifically, we define an independent Gaussian prior over parameter θ_i. Each such Gaussian has an independent variance governed by a precision hyperparameter α_i. The values for α_i are determined iteratively by maximizing the marginal likelihood function after integrating out $\boldsymbol{\theta}$. Oftentimes, the Gamma distribution is allocated to the hyperparameter $\boldsymbol{\alpha}$ to promote a light tailed $\boldsymbol{\theta}$ prior [77, 78]. Through this optimization process, some α_i values may become infinitely large, causing their corresponding parameter vectors θ_i to approach zero (the posterior distribution converges to a delta function at zero), resulting in a sparse solution. The θ_i that remains finite are considered relevant for modeling the data distribution. Thus, the Bayesian approach naturally balances enhancing the data fit and simplifying the model by diminishing the influence of certain θ_i. We explore the ARD on turbulence closure problem in Paper A [47].

Exact inference Sometimes, an appropriate choice of prior can also render the inference problem tractable, remedying the need for approximation, termed as *exact inference*. In particular, if the prior is *conjugate* to the likelihood, with an analytically tractable likelihood (e.g., no dependence on implicit FE solver), the posterior will be analytically tractable. In general, this will be the case when the prior and likelihood are from the same parametrized family e.g., the exponential family. For example, a Gaussian prior combined with aN analytically tractable Gaussian likelihood leads to a Gaussian posterior as well, the expression for which can be obtained in closed form.

Hierarchical probabilistic models and Probabilistic graphical models In many scientific and engineering applications, the data is often collected from multiple sources, each with its own set of uncertainties. Hierarchical probabilistic models [79, 80] are a powerful tool for modeling such complex data structures. In a hierarchical model, the data is assumed to be generated from a series of conditional distributions, each conditioned on the parameters of the level above it, i.e., all unknown parameters are treated probabilistically thus following a *fully* Bayesian approach. This structure allows for the incorporation of prior knowledge about the data-generating process at multiple levels of abstraction. Hierarchical models are particularly useful when the data is sparse or noisy, as they can help to regularize the estimation process and improve the robustness of the results.

Furthermore, hierarchal models can be conveniently represented by *probabilistic graphical models* (PGMs) [59, 81]. PGMs are a powerful tool for representing complex probabilistic relationships between variables compactly and intuitively. They provide a graphical representation of the dependences between variables in a model, making it easier to understand and reason about the model structure. In a PGM, nodes represent random variables, and edges represent probabilistic dependencies between variables. The structure of the graph encodes the conditional independence relationships between variables, which can be exploited to simplify the computation of the posterior distribution. A particular type of PGM expedient for our discussion is given by a Bayesian network which can be represented as a directed acyclic graph (DAG), encoding conditional independence assumptions. For instance and following the previous discussions and notations, the joint distribution between the observables $\hat{\boldsymbol{y}}$, the parameters $\boldsymbol{\theta}$ and the hyperparameter $\boldsymbol{\alpha}$ (precision to the ARD prior) can be given as

$$p(\hat{\boldsymbol{y}}, \boldsymbol{\theta}, \boldsymbol{\alpha}) = p(\hat{\boldsymbol{y}}|\boldsymbol{\theta}, \boldsymbol{\alpha})p(\boldsymbol{\theta}|\boldsymbol{\alpha})p(\boldsymbol{\alpha}). \tag{2.6}$$

The corresponding graphical model is given in Figure 2.3. Expectation Maximization [82] algorithm (see Section 2.1.9) is widely adopted for the hierarchical probabilistic models. While graphical models and corresponding specialized inference algorithms have a rich history, we will refer the reader to [81, 83].

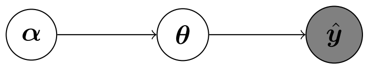


Figure 2.3: Probabilistic graphical model representing the joint probability distribution over the three random variables. The nodes shaded grey are the observed variables and nodes in no shade are the *latent*/unobserved variables.

Kullback-Leibler divergence In the realm of probabilistic reasoning and machine learning, it becomes essential to consider how one might quantify the information contained within probability distributions, or evaluate the discrepancy of information encoded between two distinct distributions, $p(x)$ and $q(x)$ for a random variable x. The Kullback-Leibler (KL) divergence [84] is a widely adopted measure of the information lost when $q(x)$ is used to approximate $p(x)$. Let us consider two densities $p, q \in \mathcal{P}$ with $\mathcal{P}$ the set of all admissible probability density functions (with assumed identical support), the divergence measure $D[\cdot\|\cdot] : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}_0^+$, then the KL divergence is defined as

$$D_{\mathrm{KL}}\left(p\|q\right) = -\int p(x)\log q(x)\,\mathrm{d}x - \left(-\int p(x)\log p(x)\,\mathrm{d}x\right) \;= \mathbb{E}_{p(x)}\left[\log\frac{p(x)}{q(x)}\right]. \tag{2.7}$$

The KL divergence satisfies the following:

- $D_{\mathrm{KL}}\left(p\|q\right) \geq 0$ with equality if and only if $p(x) = q(x)$ almost everywhere.
- The KL divergence is not symmetric, i.e., $D_{\mathrm{KL}}\left(p\|q\right) \neq D_{\mathrm{KL}}\left(q\|p\right)$.
- The KL divergence is not a metric, i.e., it does not satisfy the triangle inequality.

Although KL divergence does not define a proper metric in the general case (due to lack of symmetry and triangle equality), it can nonetheless be regarded as a statistical distance, i.e., as quantifying the discrepancy of information encoded in p and q. In conjugation with *Jensen's inequality* [85], the KL divergence is a key tool in probabilistic machine learning as we will see in the following sections and Paper A [47] and Paper E [51].

Entropy and information The entropy provides a measure of the uncertainty associated with a random variable $x \sim p(x)$. For both discrete and continuous variables, the entropy is defined as

$$H[p] = \mathbb{E}_{p(x)}[-\log p(x)]. \tag{2.8}$$

For continuous variables, this is also called the *differential entropy*. Correspondingly, the entropy collapses to zero for degenerate distributions which place their entire probability mass on a single outcome, and the entropy increases as probabilities are assigned more equally to all possible values of x. Another way to see entropy as a measure of uncertainty in a random variable is

$$H[p] = D_{\mathrm{KL}}\left(p\|u\right) + \text{const.}, \tag{2.9}$$

where u is a uniform distribution. Since $D_{\mathrm{KL}}\left(p\|u\right) \geq 0$, the less like a uniform distribution p is, the smaller will be the entropy. Since the uniform distribution contains the least information a priori about which state $p(x)$ is in, the entropy is therefore a measure of the a priori uncertainty in the state occupancy. For a discrete distribution, the entropy is positive, whereas the differential entropy can be negative.

The *cross-entropy* is a related concept, which measures the average number of bits needed to encode the outcomes of a random variable x when we use a model $q(x)$ to approximate the true distribution $p(x)$. The cross-entropy ultimately equates to the previously introduced KL divergence. Classical loss functions (e.g. discriminative classification problems) in machine learning often entail cross-entropy, as it defines the only non-constant term of the Kullback-Leibler divergence in the context of fixed empirical data distributions. Entropy and the related concepts pertain to the field of information theory, pioneered by Shannon in 1948 [86]. The field has since found applications in a wide range of fields, including machine learning, statistics, and physics. The concepts of entropy and information are fundamental to understanding the behavior of probabilistic models and the trade-offs between model complexity and generalization. For a more detailed discussion, the reader is referred to [57].

2.1.5 Approximate Inference

The bottleneck of the evidence is the central problem in the Bayesian Inference. For most probabilistic models, we will not be able to compute marginals or posteriors exactly, so we must resort to using *approximate inference* [87]. There are many different algorithms, that trade off speed, accuracy, simplicity, and generality, see for example Figure 2.4. These algorithms are reliant on numerical optimization strategies. Owing to the recent boom in ML, both approximate inference and numerical optimization have seen substantial advances [46,88]. Recently, there has been the emergence of probabilistic programming and associated numerical frameworks [89,90], which seek to provide a more automated construction of probabilistic models and execution of (approximate) probabilistic inference. In the following sections, we will provide a broad overview and limit ourselves to discussing approximate inference strategies relevant to the work presented in the thesis.

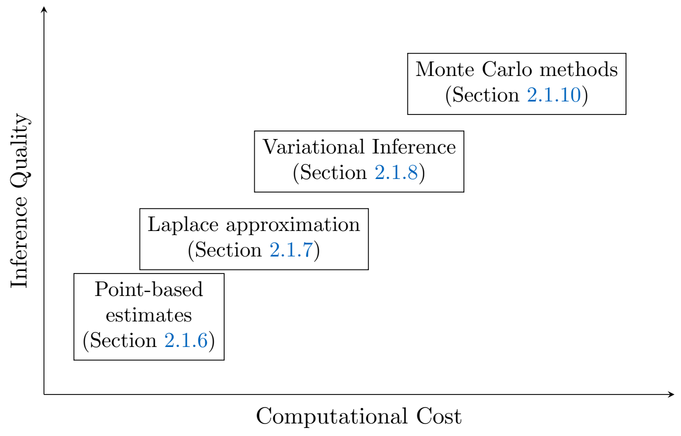

Inference Quality
Monte Carlo methods (Section 2.1.10)
Variational Inference (Section 2.1.8)
Laplace approximation (Section 2.1.7)
Point-based estimates (Section 2.1.6)
Computational Cost

Figure 2.4: Comparision of commonly used approximate inference methods based on computational cost and inference quality. Note, the figure doesn't explicitly point toward a linear relationship.

2.1.6 Point-based approximation

Point-based approximations, such as *maximum likelihood estimate* (MLE) and *maximum a posteriori* (MAP), are one of the most rough approximations to Bayesian inference [91]. The MLE is a point estimate of the parameters that maximizes the likelihood function, while the MAP estimate is a point estimate that maximizes the posterior distribution. The MLE is given by

$$\boldsymbol{\theta}^*_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta}). \tag{2.10}$$

However, the MLE does not account for the prior information, which can lead to overfitting, especially for a small number of samples. The MAP estimate, on the other hand, incorporates the prior information and is given by

$$\boldsymbol{\theta}^*_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}) = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}). \tag{2.11}$$

It's very common to operate in the log-space, as the log-likelihood is often easier to work with. So the MLE and MAP estimates are often computed as

$$\boldsymbol{\theta}^*_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta}), \quad \boldsymbol{\theta}^*_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}). \tag{2.12}$$

and then it is assumed that the posterior puts 100% of its probability on this single value:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*_{\text{MAP}}), \tag{2.13}$$

where $\delta(\cdot)$ is the Dirac delta function. The MAP estimate is often used in practice when the full posterior distribution is intractable and can be seen as a compromise between the MLE and the full Bayesian posterior, as it incorporates the prior information while still providing a point estimate. Additionally, the MAP estimate doesn't need to compute the normalization constant of the posterior, the evidence term, as it doesn't depend on the parameter $\boldsymbol{\theta}$. However, MAP has a subtle drawback. It is not invariant to the *reparametrization* of probability distributions. The MLE does not suffer from this since the likelihood is a function, not a probability density. Bayesian inference does not suffer from this problem either, since the change of measure is taken into account when integrating over the parameter space.

The advantage of point-based estimates is they can benefit from the variety of optimization algorithms available [92], even more so due to the push of the ML community. However, the point-based estimates are not able to capture the full uncertainty in the parameters, which is a key feature of the Bayesian approach. The point-based estimates are also sensitive to the choice of the prior, and the MAP estimate can be biased towards the prior.

2.1.7 Laplace approximation

Laplace approximation [93, 94] is a method for approximating the posterior distribution with a multivariate Gaussian distribution centered around the mode of the posterior. The Laplace approximation is based on the second-order Taylor expansion of the log-posterior around the mode. It approximates the posterior as

$$p(\boldsymbol{\theta}|\mathcal{D}) \approx q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}^*_{\text{MAP}}, \boldsymbol{H}^{-1}), \tag{2.14}$$

where $\boldsymbol{H}$ is the Hessian of the log-posterior evaluated at the MAP estimate $\boldsymbol{\theta}^*_{\text{MAP}}$. So it is a two-step procedure, firstly an optimization procedure to find the MAP estimate, and then approximate the curvature of the posterior at that point based on the Hessian. Compared to point-based approximations, such as MAP, the Laplace approximation also estimates the underlying uncertainties around the $\boldsymbol{\theta}^*_{\text{MAP}}$. The only additional computational cost is the computation of the Hessian (or the approximation of it) at $\boldsymbol{\theta}^*_{\text{MAP}}$. The Laplace approximation is a good choice when the posterior is unimodal and the mode is a good representation of the posterior. However, the Laplace approximation can be inaccurate when the posterior is multimodal or has long tails. For example, when the posterior is skewed and the parameter lies in a constrained interval, the Laplace approximation will be inaccurate. Fortunately, we can solve this latter problem by using a change of variable.

2.1.8 Variational Inference

In this section, we discuss variational inference (VI) [87], also known as Variational Bayes. VI is an optimization-based method used for posterior inference that offers substantial modeling flexibility, potentially leading to more precise approximations. The technique focuses on approximating complex and often intractable probability distributions, such as the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ with a tractable approximate distribution $q(\boldsymbol{\theta})$. This is particularly useful in cases when the likelihood function involves physical models, as the likelihood would not be analytically tractable. The goal is to find the best approximation $q(\boldsymbol{\theta})$ that minimizes some discrepancy D between the true posterior $p(\boldsymbol{\theta}|\mathcal{D})$ and the approximate distribution $q(\boldsymbol{\theta})$ given by

$$q^* = \underset{q \in \mathcal{Q}}{\arg\min}\, D(q, p), \tag{2.15}$$

where $\mathcal{Q}$ is some tractable family of distributions, e.g., fully factorized distributions. We optimize over the parameters $\boldsymbol{\xi}$ (also known as the variational parameters) of the approximate distribution q rather than optimizing over the functions. The discrepancy D is often the Kullback-Leibler divergence (Section 2.1.4), which in this context is given by

$$D(q,p) = D_{\mathrm{KL}}\left(q(\boldsymbol{\theta}|\boldsymbol{\xi}) \| p(\boldsymbol{\theta}|\mathcal{D})\right) = \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left[\log \frac{q(\boldsymbol{\theta}|\boldsymbol{\xi})}{p(\boldsymbol{\theta}|\mathcal{D})}\right]. \tag{2.16}$$

From the above and Eq. (2.4), the inference problem can be reformulated as an optimization problem:

$$\begin{aligned}
\boldsymbol{\xi}^* &= \arg\min_{\boldsymbol{\xi}} D_{\mathrm{KL}}\left(q(\boldsymbol{\theta}|\boldsymbol{\xi}) \| p(\boldsymbol{\theta}|\mathcal{D})\right) && (2.17)\\
&= \arg\min_{\boldsymbol{\xi}} \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left[\log q(\boldsymbol{\theta}|\boldsymbol{\xi}) - \log\left(\frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}\right)\right] && (2.18)\\
&= \arg\min_{\boldsymbol{\xi}} \underbrace{\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left[\log q(\boldsymbol{\theta}|\boldsymbol{\xi}) - \log p(\mathcal{D}|\boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\right]}_{-\mathcal{F}(\boldsymbol{\xi})} + \log p(\mathcal{D}) && (2.19)
\end{aligned}$$

where $\mathcal{F}(\boldsymbol{\xi})$ is the *evidence lower bound* (ELBO) or the variational free energy. The ELBO is a lower bound on the log-evidence $\log p(\mathcal{D})$, since $D_{\mathrm{KL}}(q\|p) \geq 0$, we have $\mathcal{F}(\boldsymbol{\xi}) \leq \log p(\mathcal{D})$. Therefore by maximizing the ELBO, the KL divergence is minimized, making the variational posterior q closer to the true posterior. Also note that the log-evidence $\log p(\mathcal{D})$ is a constant with respect to the parameters $\boldsymbol{\xi}$ of the approximate distribution, and hence it does not affect the optimization, so we drop it. Therefore, the optimization problem can now be written as

$$\boldsymbol{\xi}^* = \arg\max_{\boldsymbol{\xi}} \mathcal{F}(\boldsymbol{\xi}) = \arg\max_{\boldsymbol{\xi}} \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left[\log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta}|\boldsymbol{\xi})\right]. \tag{2.20}$$

The ELBO can be written in terms of $D_{\mathrm{KL}}\left(q(\boldsymbol{\theta}|\boldsymbol{\xi}) \| p(\boldsymbol{\theta})\right)$ as:

$$\mathcal{F}(\boldsymbol{\xi}) = \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left[\log p(\mathcal{D}|\boldsymbol{\theta})\right] - D_{\mathrm{KL}}\left(q(\boldsymbol{\theta}|\boldsymbol{\xi}) \| p(\boldsymbol{\theta})\right). \tag{2.21}$$

This can be seen as a trade-off between the likelihood of the data and the complexity of the approximate distribution. We can also write the ELBO as the following:

$$\mathcal{F}(\boldsymbol{\xi}) = \underbrace{\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left[\log p(\mathcal{D},\boldsymbol{\theta})\right]}_{\text{expected log joint}} + \underbrace{H(q(\boldsymbol{\theta}))}_{\text{entropy}}. \tag{2.22}$$

The above form of the ELBO can be interpreted as the expected log-joint probability of the data and the parameters plus the entropy of the approximate distribution. The entropy term acts as a regularizer, preventing the approximate distribution from collapsing to a point mass, while the first term encourages it to be a joint MAP configuration. Convergence to a local maximum of $\mathcal{F}$ is guaranteed due to the fact that the KL divergence is convex and $\mathcal{F}$ is consequently concave [59].

On choosing the form of variational posterior One may choose any form of approximate posterior (in a tractable family) depending on the problem at hand. Broadly, there are two main approaches for choosing the form of the variational posterior. In the first approach, we pick a convenient functional form, such as multivariate Gaussian, and then optimize the ELBO using gradient-based methods. A Gaussian is employed due to its tractability and the fact that it is fully characterized by its mean and covariance, in which case the variational parameter

$\boldsymbol{\xi} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. This is different from the Laplace approximation, since in VI, we optimize $\boldsymbol{\Sigma}$, rather than equating it to the Hessian. In the second approach, a common and simple choice for the approximate posterior is the *mean field* approximation, where the approximate posterior $q(\boldsymbol{\theta})$ is partitioned into disjoint groups. This factorized form of the densities corresponds to the *mean field* approximation. With an origin in statistical physics [95], the mean-field approximation assumes that the interactions between the variables are weak, and hence the variables can be treated as independent. The mean-field approximation is given by

$$q(\boldsymbol{\theta}) = \prod_{i=1}^{d_{\boldsymbol{\theta}}} q_i(\boldsymbol{\theta}_i). \tag{2.23}$$

The $q_i(\boldsymbol{\theta}_i)$ may or may not be of the same kind of probability distribution. When $d_{\boldsymbol{\theta}}$ is the size of $\boldsymbol{\theta}$, the mean-field approximation is also known as the *fully factorized* approximation. When it is not, it is known as the *structured mean-field* approximation. This factorization decision is usually based on the specifics of the problem at hand. The mean-field approximation is a good choice when the posterior is approximately factorizable, and the computational cost is a concern. However, it can be inaccurate when the posterior is highly correlated or multimodal. For example, if q uses a diagonal covariance matrix (corresponding to the mean field approximation), we see that the approximation is overconfident (i.e., narrow posterior variance), which is a well-known flaw of variational inference, due to the mode-seeking nature of minimizing $D_{\mathrm{KL}}(q\|p)$ [61].

Recently, *normalizing flows* [96] have gained popularity in VI, as they provide a flexible way to model complex distributions, thus providing a more accurate posterior approximation than *Gaussian VI'*. The idea is to transform a simple distribution, such as a Gaussian, into a more complex distribution by applying a series of invertible transformations.

Challenges in Optimization Due to Gradient Availability The ELBO is maximized using numerical optimization strategies which relies on the efficient computation of the gradients, the gradients of the ELBO with respect to the variational parameters $\boldsymbol{\xi}$ in this case. Unfortunately, the gradient is hard to compute since

$$\begin{aligned}\nabla_{\boldsymbol{\xi}}\mathcal{F}(\boldsymbol{\xi}) &= \nabla_{\boldsymbol{\xi}}\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left[\log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta}|\boldsymbol{\xi})\right] \\ &\neq \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}\left[\nabla_{\boldsymbol{\xi}}\log p(\mathcal{D}|\boldsymbol{\theta}) + \nabla_{\boldsymbol{\xi}}\log p(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\xi}}\log q(\boldsymbol{\theta}|\boldsymbol{\xi})\right],\end{aligned} \tag{2.24}$$

i.e., the gradient operator does not commute with the expectation operator in this case. However, to circumvent this issue, we have two main approaches. The first approach is the *reparametrization trick* [97], which allows us to write the expectation as an expectation over a simple distribution, such as a Gaussian. The trick is to rewrite the random variable $\boldsymbol{\theta} \sim q(\boldsymbol{\theta}|\boldsymbol{\xi})$ as some differentiable and invertible transformation g of another random variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, which does not depend on parameters $\boldsymbol{\theta}$, i.e., $\boldsymbol{\theta} = g(\boldsymbol{\epsilon}, \boldsymbol{\xi})$. For example, considering the case of Gaussian variational posterior with $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$, where $\boldsymbol{L}$ is the lower triangular matrix after Cholesky decomposition. Then we can write $\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{L}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$. This allows us to write the expectation as

$$\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}[\tilde{\mathcal{F}}(\boldsymbol{\theta})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[\tilde{\mathcal{F}}(g(\boldsymbol{\epsilon}, \boldsymbol{\xi}))], \tag{2.25}$$

with $\tilde{\mathcal{F}}(\boldsymbol{\theta}) = \log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta}|\boldsymbol{\xi})$. Hence the gradients are given as

$$\nabla_{\boldsymbol{\xi}}\mathcal{F}(\boldsymbol{\xi}) = \nabla_{\boldsymbol{\xi}}\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})}[\tilde{\mathcal{F}}(\boldsymbol{\theta})] = \nabla_{\boldsymbol{\xi}}\mathbb{E}_{p(\boldsymbol{\epsilon})}[\tilde{\mathcal{F}}(g(\boldsymbol{\epsilon}, \boldsymbol{\xi}))] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\xi}}\tilde{\mathcal{F}}(g(\boldsymbol{\epsilon}, \boldsymbol{\xi}))]. \tag{2.26}$$

With the Monte Carlo estimator, the gradient can therefore be estimated as:

$$\widehat{\nabla_{\boldsymbol{\xi}} \mathcal{F}(\boldsymbol{\xi})} \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\xi}} \tilde{\mathcal{F}}(g(\boldsymbol{\epsilon}^{(s)}, \boldsymbol{\xi})), \quad \text{with} \;\; \boldsymbol{\epsilon}^{(s)} \sim p(\boldsymbol{\epsilon}). \tag{2.27}$$

This is called the reparametrized gradient or the *pathwise derivative* [98]. Usually, this gradient is approximated using the Monte Carlo method, which is then propagated through the computational graph using the backpropagation algorithm. To this end, any kind of stochastic optimizers, such as Adam [88] or RMSprop [99] can be used.

The second approach is called *blackbox variational inference* (BBVI) [100, 101] which estimates the gradient by the *score function estimator* [102], also known as the *reinforce* algorithm. The score function estimator is a general-purpose method for estimating gradients of expectations, and it can be used when the reparametrization trick is not applicable. This is relevant for the scenarios when point wise evaluation of $\tilde{\mathcal{F}}(\boldsymbol{\theta})$ is possible but the gradient evaluation is not. For example, this is very common whenever physics/engineering models are involved in the inference task. To elaborate further, in such cases, the log-likelihood term $\log p(\mathcal{D}|\boldsymbol{\theta})$ involves a physics-based model $f(\cdot)$ which acts as an observation operator, which can be evaluated at a point but the gradient of the model might not be available. The physics-based model might not be differentiable by design or model gradients/adjoints are cumbersome/non-existent. In such cases, the score function estimator can be used to estimate the gradient of the ELBO. Using the log derivative trick [103] on the Eq. (2.24), the score function estimator is given by

$$\nabla_{\boldsymbol{\xi}} \mathcal{F}(\boldsymbol{\xi}) = \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})} \left[\nabla_{\boldsymbol{\xi}} \log q(\boldsymbol{\theta}|\boldsymbol{\xi}) \left(\log p(\mathcal{D}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta}|\boldsymbol{\xi}) \right) \right]. \tag{2.28}$$

We then compute a Monte Carlo approximation to this:

$$\widehat{\nabla_{\boldsymbol{\xi}} \mathcal{F}(\boldsymbol{\xi})} \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\xi}} \log q(\boldsymbol{\theta}^{(s)}|\boldsymbol{\xi}) \left(\log p(\mathcal{D}|\boldsymbol{\theta}^{(s)}) + \log p(\boldsymbol{\theta}^{(s)}) - \log q(\boldsymbol{\theta}^{(s)}|\boldsymbol{\xi}) \right), \tag{2.29}$$

The score function estimator is unbiased but has high variance, and it requires a large number of samples (standard error in MC estimate is $\mathcal{O}(1/\sqrt{S})$) to obtain an accurate estimate of the gradient. Alternatively, we can use *control variates* or variance reduction techniques to reduce the variance of the gradient estimate. The choice of the variance reduction technique depends on the problem at hand and the computational resources available. This is a vast area of research and we refer the reader to [69, 87] for a more detailed discussion of control variates.

Many probabilistic programming libraries these days support reparametrizable distributions and can automatically handle the reparametrization trick. For example, Pyro [89] and TensorFlow Probability [104] provide a wide range of reparametrizable distributions and automatic differentiation tools to compute the gradients of the ELBO. In the backend, they construct a computational graph [105] of the probabilistic model and accordingly employ e.g., reparametrization/BBVI, construct the gradient estimate and perform the optimization. Availability of gradients and estimating them is a central issue in ML and optimization tasks involving engineering/physics-based models, and researchers have proposed several strategies to overcome this. A seamless fusion of the paradigms of *probabilistic programming* [90] and *differentiable programming* [36] is the need of the hour to accelerate scientific discovery by capitalizing on the advances in machine learning [6]. In this thesis, we tried to address the aforementioned issue in Paper A [47], Paper D [50] and Paper E [51]. In particular, we propose adjoint based strategies to deal with gradient issues in Paper A [47] to enable an inference task, and in Paper E [51], we propose a general-purpose algorithm to perform black-box optimization. The gradient issue is also discussed in Section 2.2.10 and more details can be found in [36, 103, 105].

Stochastic VI For the dataset $\mathcal{D}$, the log likelihood of the data is given by $\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\boldsymbol{y}_n|\boldsymbol{\theta})$, where N is the number of data points. For large datasets, the ELBO can be computationally expensive to compute, as it requires evaluating the log-likelihood for each data point. In such cases, we can use stochastic optimization methods to approximate the ELBO. The idea is to use a mini-batch of data points to compute an unbiased estimate of the ELBO. The stochastic ELBO is given by

$$\mathcal{F}(\boldsymbol{\xi}) = \frac{N}{M} \sum_{m=1}^{M} \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\xi})} \left[\log p(\boldsymbol{y}_m|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta}|\boldsymbol{\xi}) \right], \tag{2.30}$$

where M is the mini-batch size. The stochastic ELBO is an unbiased estimate of the true ELBO, and as the mini-batch size increases, the estimate becomes more accurate. Stochastic Variational Inference (SVI) [106] is a powerful tool for scaling up variational inference to large datasets, and it is widely used in practice. The optimization of the stochastic ELBO can be done using stochastic gradient descent (SGD) or its variants, such as Adam [88] or RMSprop [99]. The choice of the optimizer and learning rate is crucial for the convergence of the algorithm. The learning rate should be chosen carefully to ensure that the optimization converges to a local maximum of the ELBO. The optimization can be further improved by using techniques such as learning rate schedules, early stopping, and momentum. We explore the application of SVI with suitable adjustments to the turbulence closure problem in Paper A [47].

In summary, we have motivated the VI, discussed more important concepts broadly and introduced algorithms relevant to the thesis. Note that the VI has a very rich literature and is a rapidly evolving field. For more details, the interested reader is directed to reviews [87, 107], and books e.g., [59, 61, 108].

2.1.9 Expectation Maximization

The *Expectation-Maximization* (EM) [82] is an algorithm designed to compute the MLE or MAP parameter estimate for probability models that have missing data and/or hidden/latent variables (as in Figure 2.3). The EM algorithm is widely used in machine learning and statistics for estimating the parameters of probabilistic models, such as mixture models, hidden Markov models, and Gaussian mixture models. Following the preceding notations, suppose we have a latent variable model of the form

$$p(\hat{\boldsymbol{y}}_{1:N}, \boldsymbol{z}_{1:N}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\hat{\boldsymbol{y}}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) p(\boldsymbol{z}_n|\boldsymbol{\theta}), \tag{2.31}$$

where $\boldsymbol{z}_{1:N}$ are the latent variables (see Figure 2.5). The EM algorithm is used to maximize the likelihood of the observed data $\hat{\boldsymbol{y}}_{1:N}$ with respect to the model parameters $\boldsymbol{\theta}$. Since the latent variables $\boldsymbol{z}_n$ are hidden, we marginalize them out to get the per sample log marginal likelihood:

$$\log p(\hat{\boldsymbol{y}}_n|\boldsymbol{\theta}) = \log \int p(\hat{\boldsymbol{y}}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) p(\boldsymbol{z}_n|\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{z}_n. \tag{2.32}$$

Unfortunately, computing this integral is usually intractable. Fortunately, the ELBO is its lower bound:

$$\log p(\hat{\boldsymbol{y}}_n|\boldsymbol{\theta}) \geq \mathcal{F}(\boldsymbol{\xi}_n, \boldsymbol{\theta}) = \mathbb{E}_{q_{\boldsymbol{\xi}_n}(\boldsymbol{z}_n)} \left[\log p(\hat{\boldsymbol{y}}_n|\boldsymbol{z}_n, \boldsymbol{\theta}) + \log p(\boldsymbol{z}_n|\boldsymbol{\theta}) - \log q_{\boldsymbol{\xi}_n}(\boldsymbol{z}_n) \right]. \tag{2.33}$$

For all the samples, we can thus perform the following optimization:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathcal{F}(\boldsymbol{\xi}_n, \boldsymbol{\theta}). \tag{2.34}$$

To this end, the EM algorithm performs the optimization by alternating between the following two steps:

- **E-step:** Fix $\boldsymbol{\theta}$ and maximize the ELBO w.r.t. the variational parameters $\boldsymbol{\xi}_n$ to give us $q_{\boldsymbol{\xi}_n}(\boldsymbol{z}_n)$.
- **M-step:** Maximize the ELBO (using the new $\boldsymbol{\xi}_n$) w.r.t. the model parameters $\boldsymbol{\theta}$.

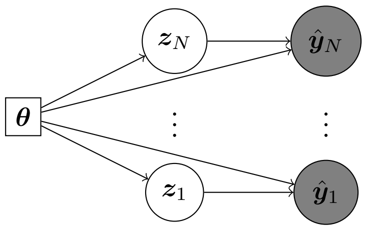


Figure 2.5: Graphical model with latent random variables $\boldsymbol{z}_{1:N}$ and observed data $\hat{\boldsymbol{y}}_{1:N}$. The *global* deterministic parameter $\boldsymbol{\theta}$ is given with rectangles, the *local* latent variables $\boldsymbol{z}_{1:N}$ are given with circles, and the observed data $\hat{\boldsymbol{y}}_{1:N}$ are given with shaded circles. The arrows indicate the dependencies between the variables.

The traditional EM algorithm assumes the posterior to be analytically tractable, as well as generally the required expectations with respect to this distribution. The *Variational-Bayes Expectation-Maximization* (VB-EM) algorithm is the generalization of the EM algorithm, where instead of being able to identify $q_{\boldsymbol{\xi}_{1:N}}(\cdot)$ in the E-step in closed form as the posterior, we conduct approximate inference (e.g., VI/MCMC) to approximate the posterior. This is possible as the EM algorithm permits incomplete or sparse updates [109], i.e., none of the two optimization problems implied by the E-step and M-step need to be solved fully in each iteration. In most of the real world cases with the likelihood involving physics-based models, the E-step is intractable, and hence the VB-EM algorithm is used. We explore the application of the VB-EM algorithm in Paper C [49].

2.1.10 Sampling-based Inference

While VI is fast, it can lead to an approximate posterior that is overconfident and potentially biased, since it is restricted to a specific functional form $q \in \mathcal{Q}$. A more flexible class of algorithms are the sampling-based inference algorithms which in general provide a better quality inference than the VI, but at the cost of higher computational complexity (see Figure 2.4). The sampling-based inference algorithms use a non-parametric approximation in terms of a set of samples. The key issue is creating the posterior samples $\boldsymbol{\theta}^s \sim p(\boldsymbol{\theta}|\mathcal{D})$ efficiently without having to evaluate the normalization constant $p(\mathcal{D}) = \int p(\boldsymbol{\theta}, \mathcal{D})\, \mathrm{d}\boldsymbol{\theta}$.

Non-iterative Monte Carlo methods, including rejection sampling and importance sampling, which generate independent samples from some target distribution can be adopted for the inference. The trouble with these methods is that they often do not work well in high dimensional spaces [110]. Therefore, a very popular sampling based inference algorithm in Bayesian statistics and machine learning is the *Markov Chain Monte Carlo* (MCMC) [110]. MCMC methods are a class of algorithms that generate samples from the target density function (e.g., the posterior distribution in Bayesian statistics) by constructing a *Markov chain*[3] that has the target density

[3]For a random variable $\boldsymbol{x}_{1:T}$, $\boldsymbol{x}_t$ captures all the relevant information about the state of the system, i.e., $p(\boldsymbol{x}_{1:T}) = p(\boldsymbol{x}_1)\prod_{t=2}^{T} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$

function as its stationary distribution. A widely adopted MCMC algorithm is the *Metropolis-Hastings (MH) algorithm* [111]. It is a general-purpose algorithm that can be used to sample from any distribution, provided that the proposal distribution is chosen appropriately. The basic idea behind MH is as follows: we start at a random point in parameter space, and then perform a random walk, by sampling new states (parameters) from a proposal distribution $q(\boldsymbol{\theta}'|\boldsymbol{\theta})$. If q is chosen carefully, the resulting Markov chain distribution will satisfy the property that the fraction of time we visit each point in space is proportional to the posterior probability. The key point is that to decide whether to move to a newly proposed point $\boldsymbol{\theta}^{'}$ or to stay in the current point $\boldsymbol{\theta}$, we only need to evaluate the unnormalized density ratio

$$\frac{p(\boldsymbol{\theta}|\mathcal{D})}{p(\boldsymbol{\theta}'|\mathcal{D})} = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})/p(\mathcal{D})}{p(\mathcal{D}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')/p(\mathcal{D})} = \frac{p(\mathcal{D},\boldsymbol{\theta})}{p(\mathcal{D},\boldsymbol{\theta}')}. \tag{2.35}$$

As can be observed, this bypasses the need to compute the normalization constant $p(\mathcal{D})$, which as discussed earlier is often intractable. The algorithm needs as input the function that computes the (log) joint density $p(\boldsymbol{\theta},\mathcal{D})$ (e.g., the log-likelihood involving the forward model $f(\cdot)$ and the prior.) and the *proposal* distribution $q(\boldsymbol{\theta}'|\boldsymbol{\theta})$ which decides which state to visit next. Commonly, Gaussian proposal distribution is used, i.e., $q(\boldsymbol{\theta}'|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}'|\boldsymbol{\theta},\sigma\boldsymbol{I})$, this is called the *random walk Metropolis* algorithm, where σ is the standard deviation of the proposal distribution. The choice of the proposal distribution is crucial for the convergence of the algorithm. If the proposal distribution is too narrow, the algorithm will take a long time to explore the parameter space, while if the proposal distribution is too wide, the algorithm will accept a large number of proposals, leading to a high rejection rate. There are strategies to counter this dependence of the proposal distribution (via adaptively adjusting the σ for example), such as the *adaptive Metropolis* algorithm [112], the *delayed rejection* algorithm [113] and *Delayed Adaptive Rejection Metropolis* (DRAM) [114](we explore its application in Paper C [49]).

However, high-dimensional distributions or distributions with complex geometries can be an issue. In such cases, more advanced MCMC algorithms, such as *Hamiltonian Monte Carlo (HMC)* [115] and *Gibbs sampling* [116], can be used. The HMC relies on the gradient computation of the log joint $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta},\mathcal{D})$ (given the unknowns are continuous) to guide the proposals into the regions of space with higher probability. This makes the HMC a good choice for sampling from high-dimensional distributions with complex geometries. The *No-U-Turn* (NUTS) algorithm [106] is an extension of the HMC algorithm that automatically tunes the step size and the number of leapfrog steps, making it more efficient than the HMC algorithm. Due to its speed and ability to handle high-dimensional, the NUTS algorithm is widely used in practice and is the default sampling algorithm in many probabilistic programming libraries, such as Stan [117], Pyro [89] and PyMC3 [118]. The Gibbs sampling is a good choice when models have a conditional independence structure, as it samples from the conditional distributions of the parameters given the other parameters. In cases when the data is arriving in a continual, unending stream, as in state-space models, *Sequential Monte Carlo* (SMC) [119, 120] can be used. SMC performs inference using a sequence of different distributions, from simpler to more complex, with the final distribution being equal to the target posterior. The SMC algorithm is widely used in practice for inference in state-space models and other time-series models. The choice of the sampling-based inference algorithm depends on the problem at hand and the computational resources available. In this section, we tried to provide a broad perspective and further details can be found in [69, 110]. An interactive visualization of many of these algorithms in 2d can be found here[4].

[4]http://chi-feng.github.io/mcmc-demo/app.html

2.1.11 Summary

We introduced various sources and types of uncertainties, introduced the paradigm of UQ, and briefly discussed uncertainty propagation. After that, we provided a broad overview of the Bayesian Inference, discussed Variational inference and sampling-based inference in brief and directed the reader to the relevant literature.

The choice of inference algorithms is non-trivial. As discussed in Figure 2.4, different approximate inference algorithms make different tradeoffs between speed, accuracy, generality, simplicity, etc. This makes it hard to compare them on an equal footing. One approach is to evaluate the accuracy of the approximation $q(\boldsymbol{\theta})$ by comparing it to the *true* posterior $p(\boldsymbol{\theta}|\mathcal{D})$, computed offline with an *exact* method. We are usually interested in accuracy vs speed tradeoffs, which we can compute by evaluating $D_{\mathrm{KL}}\left(p(\boldsymbol{\theta}|\mathcal{D})\|q_t(\boldsymbol{\theta})\right)$, where $q_t(\boldsymbol{\theta})$ is the approximate posterior after t units of compute time. Unfortunately, it is usually impossible to compute the true posterior. A simple alternative is to evaluate the quality in terms of its prediction abilities on out-of-sample observed data, similar to cross-validation. For methods on accessing the VI, see [121, 122], and for MCMC, see [123]. A general comparison and advice on VI and MCMC algorithms can be found in [87].

2.2 Probabilistic and Scientific Machine Learning

In this section, we aim to provide a snapshot of SciML research. We begin by briefly introducing machine learning in Section 2.2.1. Then we provide a snapshot of the major scientific tasks that can benefit from the SciML approaches. These are fundamental and essential tasks carried out across science, and they traditionally have long-standing general and domain-specific challenges associated with them. The tasks differ generally in terms of what is available e.g., governing equations, ground-truth observations, mechanistic model etc. In particular, we will briefly discuss forward simulation improvement (Section 2.2.2), inversion (Section 2.2.3), scientific/model discovery (Section 2.2.4), surrogate forward modeling (Section 2.2.5), reduced order modeling (Section 2.2.6), closure modeling (Section 2.2.7), and optimization involving physics-based models (Section 2.2.8). Then we discuss the different strategies that can be used to address these tasks in Section 2.2.9, followed by a discussion on *differentiable physics* in Section 2.2.10 which is a key enabler for many SciML approaches. For a more detailed treatment of the broad field of SciML, the reader is directed to [12–14, 19, 124].

2.2.1 Brief overview of Machine Learning

Machine learning is a field of artificial intelligence that focuses on developing algorithms that allow computers to learn from and make predictions or decisions based on data. It involves various techniques and models that enable systems to improve their performance over time without being explicitly programmed for every task. Machine learning can be broadly categorized into three types: *supervised learning*, *unsupervised learning*, and *reinforcement learning* [61]. In supervised learning, algorithms are trained on labeled data, where the input comes with corresponding output labels. The goal is to learn a mapping from inputs to outputs that can be applied to new, unseen data. Common applications include image classification [11], speech recognition [125], and predictive analytics [126]. Unsupervised learning deals with unlabeled data, aiming to discover underlying patterns or structures without predefined labels. Techniques such as clustering and dimensionality reduction are used to group similar data points or reduce the complexity of data. This approach is often used in anomaly detection, customer segmentation, and exploratory data analysis. Reinforcement learning involves training algorithms to make a sequence of decisions by interacting with an environment. The algorithm learns to achieve a goal by receiving feedback in the form of rewards or penalties. This method is widely used in robotics, game playing, and autonomous systems.

Deep Learning is a subset of machine learning that utilizes Neural Networks (NN) with multiple hidden layers to model complex, non-linear relationships and handle large, high-dimensional datasets. Physical systems often involve intricate interactions and dependencies that are challenging to capture with traditional analytical methods. DNNs can automatically learn these relationships from data, enabling them to accurately predict outcomes and identify underlying patterns. However, these methods require substantial computational resources due to the high number of parameters and huge number of data points involved. Huge data requirement serves as a major bottleneck whenever data is to be generated from expensive physics-based models. This is where scientific machine learning comes into play, which aims to combine the strengths of machine learning with the knowledge of physical laws to address these challenges, discussed in Section 2.2.9. Recent advancements in GPU technology and the availability of automatic differentiation frameworks have significantly benefited Deep Learning. Neural Networks are central to most deep learning algorithms. Initially developed to mimic the human brain, a neural network typically employs a large number of parameters and requires a substantial training dataset. The primary advantage of NNs is their flexibility, as they can act as universal approximators. To provide a brief overview, in this section, we will discuss feed-forward neural networks (FFNN), which are just a drop in the huge ocean of available Deep Learning architectures. For a more comprehensive overview, refer to the works by LeCun [127] and Goodfellow [128].

A feed-forward neural network (FFNN) is a class of NN where the connections between the nodes do not form cycles. The network consists of an input layer, one or more hidden layers, and an output layer, with each layer comprising several nodes (neurons). Mathematically, consider a FFNN with L layers. The input to the network is denoted by $\boldsymbol{x} \in \mathbb{R}^n$, where n is the number of input features, and the output is denoted by $\boldsymbol{y} \in \mathbb{R}^m$, where m is the number of output features. For each layer l (where $l = 1, 2, \ldots, L$), the following computations are performed: $\boldsymbol{h}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$ where, $\boldsymbol{h}^{(l)} \in \mathbb{R}^{n_l}$ is the pre-activation vector for layer l, $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ is the weight matrix for layer l, $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ is the bias vector for layer l, $\mathbf{a}^{(l-1)} \in \mathbb{R}^{n_{l-1}}$ is the activation vector from the previous layer, and n_l is the number of nodes in layer l, with learnable parameters joinlty denoted by $\boldsymbol{\omega} = \{\mathbf{W}, \mathbf{b}\}$. Now, an activation function is introduced to introduce non-linearity into the model. The activation function is applied element-wise to the pre-activation vector $\boldsymbol{h}^{(l)}$ to obtain the activation vector $\mathbf{a}^{(l)}$, given by $\mathbf{a}^{(l)} = \sigma(\boldsymbol{h}^{(l)})$. Here, σ is an activation function, such as the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$, the hyperbolic tangent function $\sigma(z) = \tanh(z)$, or the Rectified Linear Unit (ReLU) function $\sigma(z) = \max(0, z)$. The output of the network is given by $\boldsymbol{y} = \mathbf{a}^{(L)}$. The operations performed are exemplarily illustrated in the Figure 2.6. We note that we could also treat all parameters as random variables and thus turn the neural network into a Bayesian neural network.

Thus, the structure allows FFNNs to approximate complex, non-linear functions and is the basis for their powerful representation capabilities.

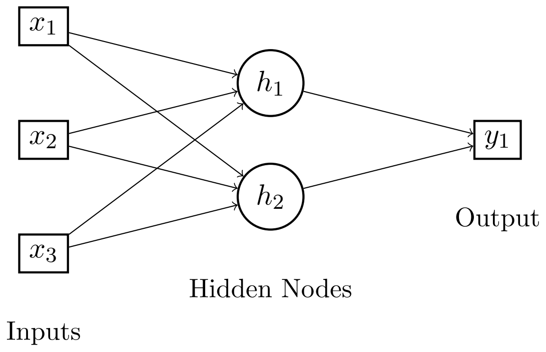


Figure 2.6: A simple feed-forward neural network.

2.2.2 Forward simulation improvement

One essential task is carrying out physics-based simulation to describe the system under study. The task essentially involves predicting certain properties of the system given some input conditions of the system. This is done by solving the governing equations of the system. For example, in the context of fluid dynamics, this would involve solving the Navier-Stokes equations.

For several scientific problems, the governing equations are given by a set of (partial) differential equations. In many cases, the system can be described by

$$\begin{aligned} \mathcal{G}[\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{\mu}] &= \mathcal{J}(\boldsymbol{x}), \quad && \boldsymbol{x} \in \Omega, \\ \mathcal{B}_k[\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{\mu}] &= b_k(\boldsymbol{x}), \quad && \boldsymbol{x} \in \Gamma_k \subset \partial\Omega, \end{aligned} \tag{2.36}$$

where $k = 1, 2, \ldots, n_b$, n_b is the number of boundary conditions, $\mathcal{G}$ is a differential operator, $\mathcal{B}_k$ is a set of of boundary operators, Ω is the domain of interest, Γ_k are the boundaries of the domain, $\boldsymbol{u}(\boldsymbol{x}) \in \mathbb{R}^{d_u}$ is the solution field with $\boldsymbol{x}$ being a d dimentional input vector in the domain Ω, $\boldsymbol{\mu}$ is a set of parameters, $\mathcal{J}(\boldsymbol{x})$ is right hand side (e.g., a source function) and b_k is a set of boundary conditions. Many different differential equations can be represented in this form, including those with time-dependence (notation corresponding to time is dropped here for simplicity. If we include it, we will have $\boldsymbol{u}(\boldsymbol{x}, t)$ and the initial conditions) or time-independence, and linear or nonlinear operators. For brevity in the downstream discussions, without loss of generality, we cast the problem given in Eq. (2.36) to the form

$$\boldsymbol{F}[\boldsymbol{u}; \boldsymbol{\varphi}] = 0, \tag{2.37}$$

where $\boldsymbol{F}$ is e.g., the PDE operator containing the spatial and temporal derivatives and $\boldsymbol{\varphi}$ represents the physical parameters or parametrized initial or boundary conditions, i.e., $\boldsymbol{\varphi} = \{\mathcal{J}(\boldsymbol{x}), b_1(\boldsymbol{x}), \ldots, b_{n_b}(\boldsymbol{x}), \boldsymbol{\mu}\}$. Comparing it to the notations introduced in Section 2.1.1, the output $\boldsymbol{y}$ takes the form of a solution of the PDE $\boldsymbol{u}$ and the function f in this case is the PDE operator $\boldsymbol{F}$. Since input parameters in Section 2.1.1 is a random variable $\boldsymbol{\theta}$, which is not always the case in this setting, we introduce a new notation $\boldsymbol{\varphi}$.

Simulating complex, multi-scale, and multi-physics systems as described by the equations above poses significant challenges, primarily due to the extensive computational resources required. These simulations demand sophisticated implementations using discretization techniques, such as finite difference, finite element, or spectral methods, and can consume millions of CPU hours on the most advanced supercomputers. To mitigate these high computational costs, several general and domain-specific strategies have been developed [129, 130]. Techniques such as adaptive mesh refinement, subgrid parameterizations, and reduced-order modeling are commonly employed. However, these methods typically necessitate a trade-off: reducing computational demands often means accepting a lower fidelity in the approximation of the physical system being simulated.

In the Section 2.2.9, we will discuss how SciML brings promise to address these challenges in the forward simulation by allowing the possibility to learn from previous simulations, providing more powerful computational shortcuts whilst having less impact on the simulation fidelity. Additionally, several real-world multiscale systems are crippled by the so-called *closure problem* (discussed in Section 2.2.7), where some quantities and processes cannot be fully prescribed despite their effects on the simulation's accuracy. SciML offers a promising solution to this problem by learning the closure terms by combining traditional (physics-based) modeling with data-driven (machine-learned) techniques.

2.2.3 Inversion

The goal of the inversion task is to estimate a set of latent parameters of a system given a set of real-world observations of the system. Following the framework given by Eq. (2.37), it amounts to estimating $\boldsymbol{\varphi}$ given $\boldsymbol{u}$ and $\boldsymbol{F}$. For example, in the context of fluid dynamics, this could involve estimating the viscosity of a fluid given some velocity/pressure observations of the flow. Inverse problems can be challenging for a number of reasons. Firstly, the problem is often ill-posed, meaning that the solution may not be unique or stable (also discussed in Section 2.1.4). Secondly, the problem may be computationally expensive, requiring many forward simulations to estimate the parameters. Finally, the problem may be high-dimensional, with many parameters to estimate.

SciML offers a number of strategies to address these challenges. For example, one could use a Bayesian approach to estimate the parameters (see Section 2.1.4), which would provide a probabilistic estimate of the parameters and quantify the uncertainty in the estimate [47, 131, 132]. Alternatively, one could use a surrogate model to approximate the forward simulation, reducing the computational cost of the inversion [26, 133, 134]. Finally, one could use a hybrid approach, combining physics-based and data-driven models to estimate the parameters [43, 47]. These approaches are discussed in more detail in the following sections.

2.2.4 Scientific/model discovery

There are many situations where we lack a complete understanding of the system itself, meaning we are uncertain how to define our physical model $\boldsymbol{F}$ given in Eq. (2.37). For instance, we still do not have an adequate model of many processes within the earth's climate system or to understand the complexities of the human brain. Learning about a system, such as by discovering its governing equations, is powerful because it can provide a general model of the system. This can be seen as a different type of inversion problem where the entire physical model, or at least its unknown parts, is inferred from observations of the system. This process is often very challenging and inherits many of the difficulties associated with inversion tasks mentioned earlier.

SciML is aiding this discovery by allowing us to automate the process and/or learn about complex processes. SciML approaches focus on discovering theories or equations that are consistent with our existing knowledge of scientific systems. Discovering governing equations from data goes back in time [135], but the recent advancements in deep learning and probabilistic modeling have made this process more efficient and accurate [136]. For example, the recently introduced framework of Sparse Identification of Non-linear Dynamics (SINDy) [137] that can select a parsimonious structure of the model of a dynamical system in an appropriate basis of feature transformations. Another example can be identifying the closure model [24] (see Section 2.2.7) for a system by learning the missing terms in the governing equations.

2.2.5 Surrogate forward modelling

The primary objective of surrogate forward modeling is to build computationally efficient ML-based surrogates of a physics-based forward model [22, 138] (see Eq. (2.37)). Although $\boldsymbol{F}$ is assumed to be accurate, the original function might be complex and specific to a particular design problem, whereas surrogate models are usually simpler and drawn from generic classes of functions, such as various neural network architectures used in deep learning [134]. The surrogate's parameters are determined through a training process to imitate the original function. The training only requires access to the mechanistic forward model $\boldsymbol{F}$ to produce synthetic simulations of the solution vector $\boldsymbol{u}$ instead of ground-truth observations of $\boldsymbol{u}$.

In deep-learning-based surrogates, automatic differentiation [46] is inherently available within the machine learning framework. Surrogates can be differentiable even if the original function is not, and their evaluation, including derivatives, is significantly faster due to vectorization and hardware parallelism in GPUs and TPUs [3]. The availability of gradients thus bypasses the issues posed due to the lack of the gradients as discussed in Section 2.2.10. However, training surrogates requires numerous evaluations of the original function, scaling with the number of design parameters. Additionally, a poorly trained surrogate can introduce bias in subsequent analyses.

2.2.6 Reduced order Modeling

Reduced-order modeling (ROM) seeks to create a simplified representation of the physics-based model $\boldsymbol{F}$ using lower-dimensional forms of the relationships between the inputs $\boldsymbol{\varphi}$ and outputs $\boldsymbol{u}$. Like surrogate modeling, it relies on synthetic data from a high-fidelity, complete model. However, rather than just replacing $\boldsymbol{F}$ with a computationally efficient surrogate, reduced-order modeling aims to uncover a reduced set of equations or relationships that capture most of the system's dynamics.

For instance, in fluid dynamics, a system might have a low-rank structure with a low-dimensional manifold within high-dimensional data. Reduced-order models exploit this to develop more manageable models for the system's spatio-temporal evolution [139]. This approach requires full knowledge of the governing equations of $\boldsymbol{F}$ to learn their reduced representations, unlike surrogate forward modeling, which does not need complete knowledge of these equations. The goals of reduced-order modeling are twofold: to enhance computational efficiency in forward modeling and to discover key relationships between inputs and outputs. In several real-world cases, ROM leads to *unclosed* terms in the equation system which is colloquially referred to as the *closure problem* (see Section 2.2.7). The unclosed terms demand further modeling, which can be done using SciML techniques. ROM has extensive literature in fluid dynamics, structural mechanics, and other fields, with applications in real-time simulations, uncertainty quantification, and optimization. For more details, the reader is directed to [124, 140].

2.2.7 Closure Modeling

Closure problems are omnipresent [24, 52]. They arise when a physical model that describes certain quantities of interest is created, which depends on quantities that are not of prime interest but whose effect cannot be neglected. This can be understood in the context of multiscale problem in which the quantities of interest are associated with large scales, but depend on quantities associated with small scales. If these small scales are not resolved, the large-scale equations are *unclosed*. A notable example can be found in numerical weather prediction, which aims to forecast tomorrow's weather based on the physical laws governing fluid dynamics and today's atmospheric conditions. Although these laws theoretically encompass all pertinent physics, in practice, certain phenomena like cloud formation occur on such small spatial and temporal scales that it is impossible to accurately resolve (simulate) them on a computer. Despite this, their impact on weather is significant.

Closure models are referred to by various names across different fields [24]. The term *closure* is most common in the fluid dynamics community which also spearheaded the research, specifically addressing the turbulence closure problem [2, 4, 141] and often known as subgrid or subgrid-scale (SGS) models. In numerical weather prediction and climate science, this concept is termed *parameterization*, with the development of accurate and stable parameterizations being a major challenge [142]. This terminology is also used in general circulation models, including those for the atmosphere and oceans [16]. In materials science, molecular dynamics, and computational biology, the concept is referred to as *coarse graining* [132, 143].

Typically the construction of closure models consists of two steps: (i) postulating a model form ansatz; and (ii) fitting/learning/inferring model parameters on the basis of the available data. Mapping the effect of small scales on the large scales involves non-trivial approximation, which is typically strongly nonlinear. Therefore machine learning methods form a promising approach to address the closure problem, given their ability to approximate complex functions. Consequently, a *hybrid approach* (ref. Section 2.2.9.3) to the closure problem has become a popular research field, involving a combination of a physics-based model (e.g., differential equations) describing the large scales and a machine-learning based model (e.g., neural network) to approximate the effect of the small scales. Such approaches fall within the realm of what is now known as scientific machine learning. Combining physics-based models with machine learning models is essentially the idea behind SciML. Additionally, when any sort of model is learned, it is important to quantify the uncertainty in the model predictions (see Section 2.1). This is particularly important in the context of closure models, where the model is used to predict quantities that are *indirectly* observed and potentially limited/sparse. Therefore, this implies adopting a probabilistic approach to the closure problem. Furthermore, owing to the high dimensional parametric space in multiscale problems, a gradient based learning scheme is often desired. This necessitates the need for *differentiable physics* to enable the flow of gradients between machine learning model and the physics-based model which is discussed in Section 2.2.10.

Mathematical Formulation To provide a formal treatment to the closure modeling, consider the physical system given by Eq. (2.37) as the *full model.* The space and time discretized version of the full model is given is referred to as the *high fidelity model.* The degree of freedom or spectral content of the solution $\boldsymbol{u}$ can be reduced, e.g., through filtering, averaging or projection via a reduction operator $\mathcal{A}$ to obtain a reduced field $\tilde{\boldsymbol{u}}$:

$$\tilde{\boldsymbol{u}} = \mathcal{A}[\boldsymbol{u}]. \tag{2.38}$$

Now the task amounts to calculating the accurate approximation to $\tilde{\boldsymbol{u}}$. Since $\boldsymbol{F}[\tilde{\boldsymbol{u}}; \boldsymbol{\varphi}] \neq 0$ (as the operators $\mathcal{A}$ and $\boldsymbol{F}$ doesn't commute), $\tilde{\boldsymbol{u}}$ is not generally a solution. A common approach is introducing a parameterized *reduced model*

$$\tilde{\boldsymbol{F}}_{\boldsymbol{\omega}}[\tilde{\boldsymbol{v}}; \boldsymbol{\varphi}] = 0, \tag{2.39}$$

which describes an approximation $\tilde{\boldsymbol{v}} \approx \tilde{\boldsymbol{u}}$ that does not depend on $\boldsymbol{u}$. Here $\boldsymbol{\omega}$ denotes parameters of parameterized functions (e.g., neural networks, polynomials, etc.) that need to be learned. The solution to Eq. (2.39) generally would not yield the reduced field $\tilde{\boldsymbol{u}}$, since $\tilde{\boldsymbol{u}}$ requires knowledge of the full solution $\boldsymbol{u}$ (ref. Eq. (2.38)). This model $\tilde{\boldsymbol{F}}_{\boldsymbol{\omega}}$ should be much faster to solve than the original high fidelity model, while retaining most of the accuracy. Naturally, the challenge now amounts to discovering an expression for $\tilde{\boldsymbol{F}}_{\boldsymbol{\omega}}$ which is non-trivial owing to the often nonlinear and chaotic behavior of the full model $\boldsymbol{F}$, thus placing the closure modeling in the setting of model discovery (also see Section 2.2.4). Introducing a commutator error $\boldsymbol{C}$, we get

$$\boldsymbol{F}[\tilde{\boldsymbol{u}}; \boldsymbol{\varphi}] + \boldsymbol{C}[\boldsymbol{u}, \tilde{\boldsymbol{u}}; \boldsymbol{\varphi}] = 0. \tag{2.40}$$

Now the commutator error $\boldsymbol{C}$ can be approximated by a closure model $\boldsymbol{C}_{\boldsymbol{\omega}}[\tilde{\boldsymbol{u}}; \boldsymbol{\varphi}]$, giving the expression of $\tilde{\boldsymbol{F}}_{\boldsymbol{\omega}}$ by

$$\tilde{\boldsymbol{F}}_{\boldsymbol{\omega}}[\tilde{\boldsymbol{v}}; \boldsymbol{\varphi}] := \boldsymbol{F}[\tilde{\boldsymbol{v}}; \boldsymbol{\varphi}] + \boldsymbol{C}_{\boldsymbol{\omega}}[\tilde{\boldsymbol{v}}; \boldsymbol{\varphi}] = 0, \tag{2.41}$$

Oftentimes, the $\boldsymbol{F}$ operator involves the divergence of the stress tensor (e.g., fluid flow and material science applications), thus the closure model takes the form $\nabla \cdot \boldsymbol{C}_{\boldsymbol{\omega}}[\tilde{\boldsymbol{v}}; \boldsymbol{\varphi}]$. For instance, in the context of the Reynolds-averaged Navier-Stokes (RANS) equations with Smagorinsky model [2], the reduced model is given by

$$\tilde{\boldsymbol{F}}_{\boldsymbol{\omega}}[\tilde{\boldsymbol{v}}; \boldsymbol{\varphi}] := \underbrace{\frac{\partial \boldsymbol{v}}{\partial t} + \nabla \cdot (\boldsymbol{v} \otimes \boldsymbol{v}) - \nabla \cdot (2\nu \boldsymbol{S}(\boldsymbol{v})) + \nabla \bar{p}}_{\boldsymbol{F}[\cdot]} - \underbrace{\nabla \cdot (\nu_t(\boldsymbol{\omega}) \boldsymbol{S}(\boldsymbol{v}))}_{\nabla \cdot \boldsymbol{C}_{\boldsymbol{\omega}}[\cdot]}, \tag{2.42}$$

where $\boldsymbol{S}(\tilde{\boldsymbol{v}} = \frac{1}{2}(\nabla \tilde{\boldsymbol{v}} + (\nabla \tilde{\boldsymbol{v}})^T)$ represents the strain rate tensor, p denotes the pressure and $\nu_t(\boldsymbol{\omega})$ represents the eddy viscosity parameterized by a NN. The goal may amount to fine-tuning the parameters $\boldsymbol{\omega}$ such that the solution $\tilde{\boldsymbol{v}}$ matches as closely as possible to mean velocity/pressure $\tilde{\boldsymbol{u}}$ obtained from the high fidelity model e.g., Large Eddy Simulation (LES)/Direct Numerical Simulation (DNS). Thus we have a hybrid approach wherein the resulting PDE consists of the *known physics* ($\boldsymbol{F}[\tilde{\boldsymbol{v}}; \boldsymbol{\varphi}]$), plus *learned physics* (neural network approximation to the closure model).

Learning For approaching the closure problem in terms of the objective function that is being minimized, two broad strategies exist [52], *a priori learning* and *a posteriori learning* (also illustrated in Figure 2.7). In a priori learning, a limited number of training solutions $\boldsymbol{u}$ are generated by solving the high fidelity model, from which $\tilde{\boldsymbol{u}}$ can be extracted by applying the filtering operator $\mathcal{A}[\cdot]$. These training solutions are then used to train the closure model $\boldsymbol{C}_{\boldsymbol{\omega}}$ by minimizing some loss function e.g., mean squared error, i.e., $\mathcal{L} := \|\boldsymbol{C}_{\boldsymbol{\omega}}[\tilde{\boldsymbol{u}};\boldsymbol{\varphi}] - \boldsymbol{C}[\boldsymbol{u},\tilde{\boldsymbol{u}};\boldsymbol{\varphi}]\|^2$. In addition, this means that a priori learning amounts to residual minimization. The main advantage of the a priori approach is the relative ease of training (no differentiable solvers needed). However, a major disadvantage of this approach is that the solution of the reduced model is not part of the error metric, so instability and drift can lead to inaccurate solutions (termed as *model-data inconsistency*) [141, 144]. This a priori approach is extensively employed in the fluid dynamics community [28–30, 48, 145, 146].

An alternative approach that addresses some of the issues of a priori learning is a posteriori learning. Here, in addition to the possibility of residual minimization of the PDE, a posteriori learning offers a second possibility, namely solution error minimization i.e., $\mathcal{L} := \|\tilde{\boldsymbol{v}}_{\boldsymbol{\omega}} - \tilde{\boldsymbol{u}}\|^2$. This involves solving the reduced order model equations $\tilde{\boldsymbol{F}}_{\boldsymbol{\omega}}$ to get the solution vector $\tilde{\boldsymbol{v}}_{\boldsymbol{\omega}}$ and the training data $\tilde{\boldsymbol{u}}$ generated from high fidelity models. This leads to a major advantage as one directly targets the accurate approximation of $\tilde{\boldsymbol{u}}$, which is typically the quantity one is interested in. For instance, in RANS, one would be more interested in an accurate representation of the mean fields, i.e., the pressure and the velocity predictions rather than an accurate Reynolds stress (the unclosed term in RANS). This approach has been shown to improve stability compared to a priori learning, thus addressing the problem of the model consistency in a priori learning [141, 144]. However, involving the solver in the training process makes the optimization problem more difficult to solve, and differentiable solvers are typically needed. In order to compute $\partial\mathcal{L}/\partial\boldsymbol{\omega}$, the Jacobian $\partial\tilde{\boldsymbol{v}}/\partial\boldsymbol{\omega}$ is needed. This requires code that solves $\tilde{\boldsymbol{F}}_{\boldsymbol{\omega}}$ to be differentiable, or that an adjoint solver is available, which provides a major implementational bottleneck. In addition to a posteriori learning, the approach is known under various other names, such as *solver-in-the-loop* [39], *end-to-end learning* [41, 42], *differentiable physics*, and *online learning* [40, 43]. The differentiable physics is further discussed in the Section 2.2.10.

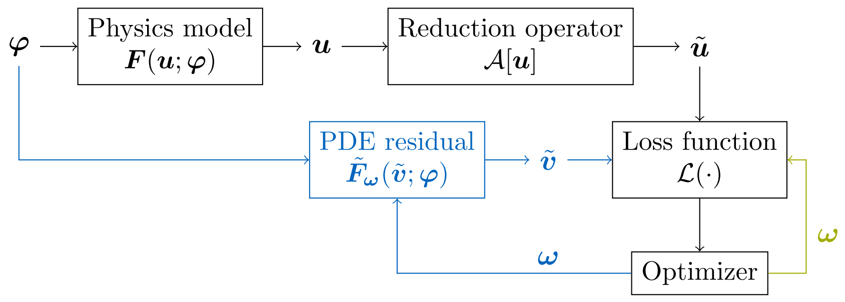


Figure 2.7: A graphical comparison between the a priori and posteriori closure modeling approaches. The output of the high fidelity physical model is passed through a reduction operator to get ground truth solution $\tilde{\boldsymbol{u}}$. In green, is the a priori learning approach, where the closure model is trained to minimize the discrepancy between the closure term $\boldsymbol{C}_{\boldsymbol{\omega}}(\tilde{\boldsymbol{u}};\boldsymbol{\varphi})$ and the *true* closure term or the commutator error $\boldsymbol{C}[\boldsymbol{u},\tilde{\boldsymbol{u}};\boldsymbol{\varphi}]$. In blue, is a posteriori learning approach, where the closure model is trained to minimize the residual of the PDE (given by Eq. (2.41)) and the discrepancy between the solution vector $\tilde{\boldsymbol{v}}$ and the ground truth $\tilde{\boldsymbol{u}}$. The optimizer updates the parameters $\boldsymbol{\omega}$ of the closure model.

Physical constraints The reduced model forms for $\tilde{\boldsymbol{F}}_{\boldsymbol{\omega}}$ vary in the extent of incorporated *known physics*. Depending on the application, further integration of physical knowledge into $\tilde{\boldsymbol{F}}_{\boldsymbol{\omega}}$ or the objective function is often possible through various possible approaches in SciML, discussed in Section 2.2.9. This further integration ensures that the physical laws like conservation and invariance are satisfied. An example is tensor-basis neural networks (TBNNs) [28, 145] for the turbulence closure problem which models the closure term as a linear expansion of strain and rotation tensors following Pope's generalized eddy viscosity hypothesis [147] while ensuring the Galleian invariance.

For more details on the closure problem and SciML strategies to address it, the reader is directed to [24, 52, 148, 149]. In Paper B [48], we use the SciML paradigm to address the turbulence closure problem and employ a-priori training and TBNN based hard constraints. In Paper A [47], we combined the three paradigms: probabilistic modeling, SciML and differentiable physics to address the turbulence closure problem. Additionally, we employ a-posteriori training and TBNN based hard constraints.

2.2.8 Optimization involving physics-based models

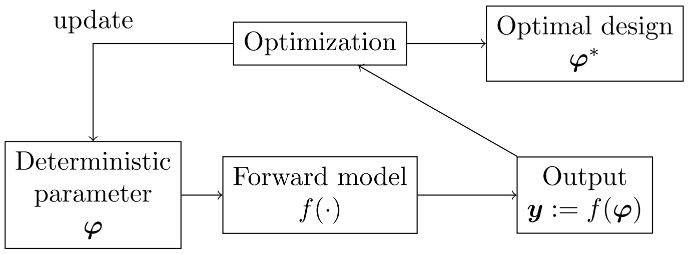
Optimization is a fundamental task in science and engineering, where the goal is to find the best design or set of parameters that optimize a given objective function. In the context of physics-based models, optimization can be used to find the optimal design of a system that minimizes drag, maximizes efficiency, or achieves some other desired outcome. Optimization problems can be deterministic or involve uncertainty, and they can be constrained or unconstrained. The optimization process can be computationally expensive, especially when the objective and constraint functions are not differentiable, as is often the case with physics-based models. SciML approaches can help address these challenges [22, 138, 150, 151] by providing efficient methods for gradient estimation, surrogate modeling, and uncertainty quantification.

Deterministic Optimization In deterministic optimization, we are interested in finding the optimal design to minimize (or maximize) a given objective (potentially involving a complex physics-based model), possibly under specific constraints. Following the notation introduced in Section 2.1.1 (with slight notational abuse), the optimization problem can be written as

$$\begin{aligned} &\min \ f(\boldsymbol{\varphi}) \\ &\text{s.t.} \quad c_i(\boldsymbol{\varphi}) \leq 0, \quad i = 1, \dots, n_c, \end{aligned} \tag{2.43}$$

where $\boldsymbol{\varphi} \in \mathbb{R}^{d_{\text{det}}}$ is the deterministic design variable (here, we assume the forward model doesn't have any stochastic inputs), $f(\boldsymbol{\varphi}) : \mathbb{R}^{d_{\text{det}}} \to \mathbb{R}$ is the scalar valued objective function, and $\{c_i : \mathbb{R}^{d_{\text{det}}} \to \mathbb{R}\}_{i=1}^{n_c}$ are the set of constraint functions. For example, considering Navier-Stokes example, the optimization problem could involve finding the optimal wing shape that minimizes drag. The optimization pipeline is graphically illustrated in Figure 2.8. As can be seen from the figure, the optimization process proceeds iteratively, updating the current best design and objective until convergence.

Extensive literature exists on various optimization approaches depending on the characteristics of the objective and constraint functions. Examples include linear optimization, convex optimization, and nonlinear optimization [92]. Most of the methods rely on derivative information regarding the objective and constraint functions. However, when the objective and constraint functions are not differentiable, as is mostly the case when any sort of physics-based model is involved, gradient-free optimization methods [152] are used.



update
Optimization
Optimal design φ^*
Deterministic parameter φ
Forward model $f(\cdot)$
Output $\boldsymbol{y} := f(\varphi)$

Figure 2.8: A graphical illustration of deterministic optimization involving physics-based models. Note that we dropped the constraint notation for ease of representation.

Optimization under uncertainty The optimization under uncertainty (OUU) attempts to answer the following question: *What is the optimal choice of deterministic inputs (of the physical model) which yields the best value for a chosen output statistic, given the inherent stochastic parameters?* Revisiting the Navier-Stokes example, the OUU problem would involve finding the optimal design (on average for example) of a wing shape that minimizes drag, given the uncertainty in the air density and velocity. The OUU can also be viewed as a generalization of deterministic optimization, where random variables are added to the optimization problem. This addition of randomness introduces complexity to the problem, which is why such problems are often referred to as stochastic optimization. The goal of OUU is to find a robust solution that remains effective despite the uncertainties present in the system, leading to the alternative term *robust optimization*. The resulting OUU problem can be formulated as

$$\begin{aligned} &\min \mathcal{R}^f(\boldsymbol{\varphi}, \boldsymbol{\theta}) \\ &\text{s.t.} \quad \mathcal{R}^{c_i}(\boldsymbol{\varphi}, \boldsymbol{\theta}) \leq 0, \quad i = 1, \dots, n_c, \end{aligned} \tag{2.44}$$

where $\boldsymbol{\theta}$ is a random variable as defined earlier in Section 2.1.1, $\mathcal{R}^f$ and $\mathcal{R}^{c_i}$ are measures of robustness or risk [62] for objective and constraint functions, respectively.

When physics-based models are used in the optimization, the uncertainty encapsulated by the random vector $\boldsymbol{\theta}$ can arise from various sources, such as model parameters, boundary conditions, or initial conditions. OUU is particularly important in engineering design, where the performance of a system can be affected by uncertainties in the environment, materials, or operating conditions. By incorporating uncertainty into the optimization process, engineers can design systems that are more reliable, efficient, and cost-effective. Assume the objective and constraints to be square integrable with respect to $\boldsymbol{\varphi}$, i.e., the variance is finite. The most commonly used robustness measure is the expected value of the objective function, which is minimized over the random variables. This leads to the formulation of the expected value optimization problem, which is a special case of OUU. The expected value optimization problem can be written as

$$\begin{aligned} &\min \mathbb{E}_{\boldsymbol{\theta}}[f(\boldsymbol{\varphi}, \boldsymbol{\theta})] \\ &\text{s.t.} \quad \mathbb{E}_{\boldsymbol{\theta}}[c_i(\boldsymbol{\varphi}, \boldsymbol{\theta})] \leq 0, \quad i = 1, \dots, n_c. \end{aligned} \tag{2.45}$$

The expected objective value yields an average-optimized design, while the set of expected constraint values indicates expected feasibility.

The OUU workflow is graphically illustrated in Figure 2.9. In the figure, on a high level, there exist three layers contributing to the computational cost which is a major challenge in OUU, more so when a physical model is involved. The first is the physical model $f(\cdot)$ itself which in most physics and engineering cases can be prohibitively expensive to carry out the OUU. In such scenarios, resorting to ROMs (Section 2.2.6) or surrogate models (Section 2.2.5) are very common. The second is the OUU layer (dashed arrow in the Figure 2.9) which benefits from fast convergence i.e., requiring least amount of optimization steps. This is contingent on the availability and the quality of gradients which can be obtained using adjoint methods or other gradient estimation techniques (if not available inherently, as is often the case with physics-based models). To address the issue of differentiability whenever physics-based models are involved in optimization/inference, the field of differentiable physics has emerged, which is also a central pillar of this thesis. We will discuss this in Section 2.2.10. There exist both gradient-based and gradient-free methods for OUU [37, 92]. The gradient-based methods are more efficient and require fewer function evaluations, but they are sensitive to the quality of the gradients. The gradient-free methods, on the other hand, are more robust but require more function evaluations. The choice of method depends on the specific problem and the computational resources available. The third layer (solid arrow in the Figure 2.9) is the uncertainty propagation layer which is also computationally expensive. This layer is responsible for propagating the uncertainty in the system through the physical model. The computational cost of this layer can be reduced by using multi-fidelity methods [68] for example. We address the second and the third layer in Paper E [51]. Summarallily, to enable the OUU with gradient-based methods, we introduce an efficient method for *well-behaved* gradient estimation coupled with multi-fidelity methods. Optimization is a vast field. For further details, the reader is directed to [36, 37, 92, 152–154].

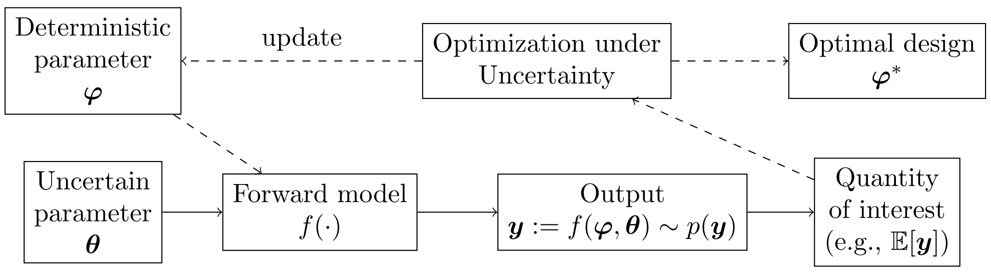

Deterministic parameter $\boldsymbol{\varphi}$
update
Optimization under Uncertainty
Optimal design $\boldsymbol{\varphi}^*$
Uncertain parameter $\boldsymbol{\theta}$
Forward model $f(\cdot)$
Output $\boldsymbol{y} := f(\boldsymbol{\varphi}, \boldsymbol{\theta}) \sim p(\boldsymbol{y})$
Quantity of interest (e.g., $\mathbb{E}[\boldsymbol{y}]$)

Figure 2.9: A graphical illustration of optimization under uncertainty. The inner loop (in solid arrow) is the forward uncertainty propagation (given in Section 2.1.3) and follows similar notation as in Figure 2.1, and the outer loop (in dashed arrow) is the optimization loop.

2.2.9 Overview of methods in SciML

In this section, we will provide a broad overview of SciML approaches and their high level categorization, but before that, we will motivate the need for the wide spectrum of algorithms. Adhering to physical laws like conservation and invariance is crucial for the widespread use of machine learning models in computational physics. Data-driven models that comply with these laws enhance interpretability, generalization, robustness, and data efficiency. Consequently, there is a concerted effort to develop algorithms with embedded constraints to avoid violating physical laws. Based on the degree to which the scientific principles are enforced, constraints in data-driven models can be *soft* or *hard*. Soft constraints use data augmentations or weak penalties in loss functions to approximate constraint satisfaction. An example is Physics Informed Neural Networks (PINNs) (discussed in Section 2.2.9.1), which penalize deviations from governing laws as a residual term in model optimization. These models are typically easier to construct and integrate into existing workflows. Conversely, *hard* constraints involve designing machine learning models to inherently satisfy constraints (see Section 2.2.9.4), an example is the tensor-basis neural networks (TBNNs) [28, 145]. Though these algorithms require more time for construction, training, and deployment, they ensure constraint satisfaction during both interpolation and extrapolation.

The methods commonly used to infuse scientific knowledge into the ML can also be categorized on a high level based on different ways to incorporate the scientific knowledge, e.g., by adding the knowledge of the governing equations in the loss function itself (see Section 2.2.9.1, Section 2.2.9.2), by adjusting the architecture of the NN so as to conform to physics (see Section 2.2.9.4), and by using hybrid methods which integrates both physics-based and ML models tightly in a hybrid fashion (see Section 2.2.9.3, and also Section 2.2.10 as the hybrid approaches mostly necessitate differentiable physics).

We will now discuss a few well-adopted methods and/or methods relevant to this thesis in the following sections. We would like to point out that we discuss only a handful of methods here, and the field is vast with many more methods and approaches that are being developed and used in practice, with different ways to categorize them. To point out a few, [155, 156] combined the probabilistic methods with SciML by introducing physical features in the loss function in a Bayesian setting. They achieved this by making the probabilistic model privy to the physical constraints by means of pseudo-observed nodes in the probabilistic graphical model, thus allowing to decide how strong the constraints are enforced by setting the relevant standard deviation of the *virtual observable*. Another recent and very promising direction is the *foundation models* [157] for scientific domains. These generalist models are *pretrained*, at-scale, on large datasets drawn from a diverse set of data distributions. They leverage the intrinsic ability of neural networks to learn effective representations from pretraining and are then deployed on a variety of downstream tasks by *finetuning* them on a few task-specific samples. Examples of foundation models for scientific domains include robotics [158], chemistry [159], biology [21], climate [15, 17] and medicine [160]. Very recently, [161] proposed a foundation model for PDEs. They claim to outperform the baselines in terms of sample efficiency and accuracy, and claim to generalize well to unseen and unrelated PDEs. For a more detailed treatment of the broad field of SciML, the reader is directed to [12–14, 19, 124].

2.2.9.1 Physics-Informed Neural Networks

Following the discussions in the surrogate modeling section (Section 2.2.5), to train the ML-based surrogate of $\boldsymbol{F}$, along with simulation data produced by $\boldsymbol{F}$, one can also leverage the governing equations of the physics-based model if it is available. This gives surrogate models that are generalizable as well as scientifically consistent. One of the promising lines of research in SciML for surrogate modeling includes the development of physics-informed neural networks (PINNs) [25] for solving PDEs in a more computationally efficient way than existing physics-based solvers.

PINNs model the solution of a PDE with a neural network and incorporate the residual of the PDE as an additional term in the loss function as a *soft* constraint, which is minimized to find the optimal parameter values. The method, therefore, can be described as a collocation method that employs a neural network as the trial function. Following the notations in Eq. (2.37), PINNs aim to model the solution $\boldsymbol{u}_{\boldsymbol{\omega}}(\boldsymbol{x})$ with a NN with parameters $\boldsymbol{\omega}$. The loss function for the PINNs is given by

$$\mathcal{L}_{\text{PINNs}} = \mathcal{L}_{\text{boundary}} + \mathcal{L}_{\text{physics}}, \tag{2.46}$$

where $\mathcal{L}_{\text{boundary}}$ is the boundary loss term that measures the discrepancy between the predicted solution and the observed data in a domain (typically at the boundaries), and $\mathcal{L}_{\text{physics}}$ is the physics loss term that measures the discrepancy between the predicted solution and the governing PDE. The physics loss term is given by

$$\mathcal{L}_{\text{physics}} = \frac{1}{N_{col}} \sum_{i=1}^{N_{col}} \|\boldsymbol{F}[\boldsymbol{u}_{\boldsymbol{\omega}}(\boldsymbol{x}_i); \boldsymbol{\varphi}]\|^2, \tag{2.47}$$

where N_{col} are the number of collocation points in the domain, $\{\boldsymbol{x}_i\}$ are the set of points sampled over the entire domain. The boundary loss term is given by

$$\mathcal{L}_{\text{boundary}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \|\boldsymbol{u}_{\boldsymbol{\omega}}(\boldsymbol{x}_i) - \boldsymbol{u}_{\text{obs}}(\boldsymbol{x}_i)\|^2, \tag{2.48}$$

where N_b is the number of data points at the boundary, $\boldsymbol{x}_i$ are the spatial coordinates of the data points, and $\boldsymbol{u}_{\text{obs}}(\boldsymbol{x}_i)$ are the observed data at the boundaries. Intuitively, the physics loss pushes the neural network to learn a solution which is consistent with the underlying differential equation, while the boundary loss attempts to ensure the solution is unique by matching the solution to the known boundary conditions. Due to its ease of implementation and ability to leverage the governing equations, PINNs have been successfully applied to a wide range of problems in scientific computing, including fluid dynamics, heat transfer, and structural mechanics to name a few [5, 130, 162, 163].

2.2.9.2 Operator learning for physics

This section discusses approaches that incorporate governing equations into their loss functions, wherein they learn an entire family of solutions by including certain inputs of the PDE (e.g., $\boldsymbol{\varphi}$ in Eq. (2.37)) as inputs to the network. This is unlike the PINNs, which only learn a single solution. These methods are particularly useful when the governing equations are known but the initial or boundary conditions are not. Consequently, they do not require retraining for new simulations and provide a fast surrogate model during inference. Mathematically, the objective is to learn an operator that maps function spaces to function spaces, rather than just a single function. More specifically, the objective is to approximate the solution operator $\mathcal{G}_{\text{sol}} : \Theta \to \mathcal{U}$ with an operator $\mathcal{G}_{\text{sol},\boldsymbol{\omega}} : \Theta \to \mathcal{U}$ parameterized by tunable $\boldsymbol{\omega}$, where Θ is the space of input functions and $\mathcal{U}$ is the space of solutions.

These approaches include algorithms that approximate a discretization, on a fixed grid of the underlying solution operator. These can be based on convolutions [164], graph neural networks [151, 165], or transformer architectures [166, 167]. Other operator learning algorithms are *neural operators* which can directly process function space inputs and outputs. These include physics-informed deep operator networks [168] which essentially conditioned PINNs on input functions of the differential equation by using the Deep Operator Network (DeepONet) architecture introduced by [26]. They showed that this approach was able to learn fast and accurate surrogate models of the solution to a nonlinear diffusion-reaction system given the source term of the equation as input, and of the solution to Burgers equation given the initial condition as input. Towards a similar goal, [27, 133] proposed Fourier Neural Operators (FNOs) that learn an operator to map between function spaces by using a series of stacked Fourier layers, where the input to each layer is Fourier transformed and truncated to a fixed number of Fourier modes. They showed that FNOs could learn the solution to the Navier-Stokes equations and the Schrödinger equation with high accuracy and efficiency.

2.2.9.3 Hybrid approaches

Hybrid approaches aim to integrate both physics-based and machine learning (ML) approaches to achieve superior predictive performance compared to models relying solely on ML or scientific methods. As introduced in [14], there are two primary types of hybrid-science-ML modeling techniques. The first type, known as *combination-based* approaches, involves simultaneously running both the science-based and ML models to generate enhanced output variable estimates with improved predictive accuracy compared to the actual observations (addressing the scientific task discussed in Section 2.2.2). Differently put, in this approach, physics-based models are fed into the ML model that finally predicts the target variables. An example of this is the long-standing practice in the scientific community of using statistical methods, often linear regression, to model the residuals of science-based models, a technique known as residual or discrepancy modeling. This method provides better predictive accuracy than using science-only or ML-only models.

The second type is *embedding-based* approaches, where ML models are intricately integrated within the framework of the science-based model to estimate intermediate quantities and parameters that are challenging to determine directly due to high computational costs. In other words, ML methods are trained to predict parameters of physics-based models, which when fed into the forward models produce better estimates of the output variables. An example of this is closure modeling (see Section 2.2.7), where ML models are used to estimate the closure terms in the governing equations of the physical model, which are often challenging to determine directly due to the high computational costs associated with resolving the small-scale physics [4, 47, 52].

The hybrid approaches are reliant on a seamless flow of gradient information between the ML model and the physics-based model, which necessitates the need for differentiable physics. Attaining differentiability is challenging, especially for the legacy solvers as it poses massive implementational effort. A typical scenario for the hybrid approach is illustrated in the Figure 2.10 in the context of differentiable physics, and more details can be found in Section 2.2.10.

2.2.9.4 Physics guided ML architecture

Physics guided ML architecture is the set of approaches that change the architecture of the ML algorithm so that it incorporates scientific constraints. Instead of treating the ML algorithm as a *black-box*, we modify its design to adhere to these constraints in a *hard* fashion. For deep learning techniques, this often involves altering the neural network architecture. Integrating scientific principles in this manner can limit the range of models the algorithm can learn, leading to more generalizable and interpretable models. From a machine learning standpoint, this introduces a strong inductive bias into the model.

There are several possible directions for building SciML architectures. We will discuss a few here. First, one can design ML architectures that encode known forms of invariances and symmetries in the predicted outputs of the system. For example, [28, 145] proposed rotationally invariant tensor basis neural networks for estimating the Reynolds stress anisotropy tensor in turbulent flows. They ensured the neural network's prediction stayed the same when the input axes of the flow were rotated, by noting that the anisotropy tensor lies on a basis of isotropic tensors and by using the neural network to predict the coefficients of the tensor in this basis. They found that this approach provided more accurate flow predictions than a generic neural network architecture. Second, in applications where scientific knowledge is available in the form of relationships between physical variables, SciML architectures can be designed to encode such knowledge in the connectivity patterns of nodes in a neural network architecture. An example of previous research in this direction includes the framework of physics-guided architecture of Long Short Term Memory (LSTM) (PGA-LSTM) models [169] that can encode monotonic relationships between density and depth of water as connections between LSTM node features in the illustrative application of lake temperature modeling. Third, SciML architectures can also be constructed where the sequence of intermediate features extracted at the hidden layers of a neural network are informed by scientific knowledge. An example of previous work in this direction is the framework of PhyNet [170] developed for modeling multi-phase flow, where the features extracted at the hidden layers of the network correspond to physically meaningful intermediates in the known scientific pathway from inputs to outputs.

2.2.10 Differentiable physics

A more potent hybrid methodology in the SciML (see Section 2.2.9.3) involves elucidating the inner workings of a conventional algorithm (e.g., physics-based models) and seamlessly embedding machine learning (ML) models within it. This integration facilitates a finer balance between the two paradigms. ML can be employed in areas where problem-solving remains ambiguous or where the conventional process is computationally intensive, while traditional components are preserved where robust and interpretable results are essential. This approach frequently enhances the performance of the traditional workflow, with the added benefit that the ML components are easier to train, more interpretable, and demand fewer parameters and less training data compared to a purely ML-driven approach.

A general approach to doing so is to make the physics-based model differentiable, i.e., colloquially known as *differentiable physics* [39, 40, 43]. We require the embedding of these physical computations and/or simulations into the probabilistic graphical model and therefore the computational graph defined by it, thus combining the paradigms of *probabilistic programming* and *differentiable programming* [46]. Quoting [36], "Differentiable programming is a programming paradigm in which complex computer programs (including those with control flows and data structures) can be differentiated end-to-end automatically, enabling gradient-based optimization of parameters in the program." In particular, the central realization of this field is that many traditional scientific algorithms can be written as a composition of basic and differentiable mathematical operations (such as matrix multiplication, addition, subtraction, etc), and that modern automatic differentiation and differential programming languages [46] make it easy to track and backpropagate the gradients of these outputs with respect to their inputs. This unlocks the possibility of inserting and training gradient-based ML components (such as neural networks, SVI, etc.) within traditional workflows, whereas otherwise it may have been difficult to do so. This is particularly important in several of the challenges discussed in the thesis and addressed by the authors work, like the closure problem (see Section 2.2.7 and Paper A [47]), Bayesian inference (see Section 2.1.4, Paper A [47] and Paper C [49]), and optimization involving physics-based based simulators (see Section 2.2.8 and Paper E [51]). A typical scenario of the need for differentiable physics is illustrated in Figure 2.10. We categorize the ways to make a physical simulator differentiable in three main ways: i) rewriting the physics-based model in a differentiable programming language like PyTorch, JAX, or TensorFlow to obtain gradients via automatic differentiation (Section 2.2.10.1), ii) the adjoint method (Section 2.2.10.2), and iii) surrogate models (Section 2.2.5). Finite differences are in principle also possible but require for the simple forward difference $n+1$ simulations for one gradient estimate with n being the dimensionality of the parameter space. This is computationally expensive and not feasible for high-dimensional parameter spaces. Recently, automatic differentiable (AD) compiler methods [171] have emerged, providing gradients of legacy codes written in C, C++, Fortran, etc. These methods are in their early stages of development and the reader is directed to [171] for more details.

2.2.10.1 Automatic Differentiation

Automatic differentiation [36,46] (or also algorithmic differentiation) essentially refers to the automated differentiation of computer programs, i.e., providing gradients and higher-order derivatives without any additional implementation overhead and ideally for a numerical cost roughly comparable to the forward pass. The elementary operations involved in automatic differentiation are represented by a directed acyclic graph (DAG). In this graph, nodes correspond to elemental operations, while edges represent the inputs and outputs between preceding and succeeding nodes. Gradient information can be acquired in multiple ways: *forward mode* automatic

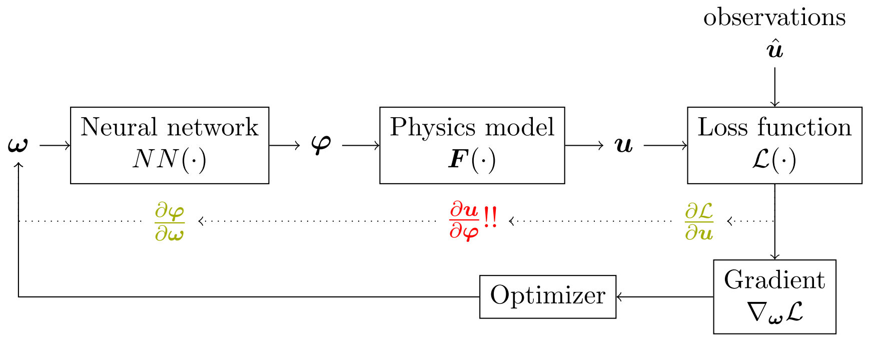

observations
$\hat{\boldsymbol{u}}$
$\boldsymbol{\omega}$
Neural network $NN(\cdot)$
$\boldsymbol{\varphi}$
Physics model $\boldsymbol{F}(\cdot)$
$\boldsymbol{u}$
Loss function $\mathcal{L}(\cdot)$
$\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{\omega}}$
$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\varphi}}$!!
$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}}$
Gradient $\nabla_{\boldsymbol{\omega}}\mathcal{L}$
Optimizer

Figure 2.10: Graphical model for the hybrid approach of SciML i.e., involving both physics models and neural networks. Notation follows from our previous discussions. The observed data is given by $\hat{\boldsymbol{u}}$. The loss function $\mathcal{L}(\cdot)$ can be e.g., MSE, ELBO, likelihood, etc. The gradient computation of the loss function w.r.t the neural network parameters $\boldsymbol{\omega}$ is usually impeded by the lack of gradients of the phsyics-based model (in red). Addressing this impediment is the goal of differentiable physics.

differentiation relies on tangent linear code, propagating linear perturbations through the model and hence directional derivatives. However, this approach becomes prohibitively expensive for higher dimensions.

More applicable to our problem is *reverse mode* automatic differentiation, which begins with a forward pass to compute all nodal values of the DAG. It then uses the linearized model to backpropagate gradient information in the form of the adjoint value. This method is analogous to the backpropagation algorithm commonly used in neural networks and the adjoint partial differential equation (PDE) approach (see discussion in the following section). The implementation of reverse mode automatic differentiation is most commonly achieved by overloading the forward operations with their corresponding backward operations. This is done by defining the forward operations in a way that they store the necessary information for the backward pass. The backward pass then uses this information to compute the gradients. This is the approach taken by modern automatic differentiation libraries like PyTorch [172], TensorFlow, and JAX. Thus, one way to bypass the gradient bottleneck posed in Figure 2.10 is to rewrite the physics-based simulator using a framework allowing automatic differentiation. However, this might pose significant implementation challenges, often involving extensive modifications to legacy codes.

2.2.10.2 The Adjoint Method

From the viewpoint of automatic differentiation we can consider the discretized solution of the partial differential equation as a node in the computational graph. Following Figure 2.10, the mapping is from the parameters of the PDE $\boldsymbol{\varphi} \in \mathbb{R}^{d_\varphi}$ to the solution of the PDE $\boldsymbol{u} \in \mathbb{R}^{d_u}$. In contrast to explicit layers such as exemplarily feedforward or convolutional layers, the mapping implied by the discretized partial differential equation is only defined *implicitely* by a numerical residual $\boldsymbol{F}(\boldsymbol{u};\boldsymbol{\varphi}) = 0$, which acts as an implicit layer [173]. We assume $\boldsymbol{F} \in \mathbb{R}^{d_u}$ from a suitable numerical discretization scheme. To this end, the objective amounts to computing gradients of some functional $\mathcal{L}(\cdot)$ involving the PDE w.r.t the parameters $\boldsymbol{\varphi}$, e.g., the functional can be the ELBO with the joint distribution involving the PDE, or the MSE.

We now consider the mathematical formulation of the adjoint method, employing the hybrid approach illustrated in the computational graph depicted in Figure 2.10. Following the notations in Figure 2.10, the gradient of the functional $\mathcal{L}$ w.r.t the parameters $\boldsymbol{\omega}$ can be computed by the chain rule as

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\omega}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\varphi}} \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{\omega}}. \tag{2.49}$$

The bottleneck in the above is the derivative of the solution of the PDE w.r.t. to the parameters, i.e., $\partial \boldsymbol{u}/\partial \boldsymbol{\varphi}$ (marked red in Figure 2.10). This is where the adjoint method comes into play. For the solution and parameter to satisfy the PDE, the residual must vanish, i.e., $\boldsymbol{F}(\boldsymbol{u}; \boldsymbol{\varphi}) = 0$. Differentiating the residual w.r.t. the parameters $\boldsymbol{\varphi}$ gives

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\varphi}} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}} \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{\varphi}} + \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\varphi}} = 0 \tag{2.50}$$

$$\implies J = \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}\boldsymbol{\varphi}} = -\left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}\right)^{-1} \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\varphi}}, \tag{2.51}$$

given $\partial \boldsymbol{F}/\partial \boldsymbol{u}$ is invertible. Now the gradient of the functional w.r.t. the parameters can be computed as

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\varphi}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}} \left[-\left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}\right)^{-1} \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\varphi}} \right]. \tag{2.52}$$

Taking the hermitian transpose of the above equation, we get

$$\left(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\varphi}}\right)^T = -\left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\varphi}}\right)^T \left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}\right)^{-T} \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}}\right)^T. \tag{2.53}$$

Let us gather the solution of the inverse Jacobian acting on a vector, and define it to be a new variable:

$$\boldsymbol{\lambda} = \left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}\right)^{-T} \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}}\right)^T. \tag{2.54}$$

$$\implies \left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}\right)^T \boldsymbol{\lambda} = \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}}\right)^T. \tag{2.55}$$

This relationship is the adjoint equation (or adjoint system) associated with the PDE $\boldsymbol{F}(\boldsymbol{u}; \boldsymbol{\varphi}) = 0$. The adjoint equation is a linear PDE that can be solved to obtain the *adjoint* vector $\boldsymbol{\lambda} \in \mathbb{R}^{d_{\boldsymbol{u}}}$. In practice, the Jacobian matrix $\left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}\right)^T$ is never stored in memory as it can be huge, instead the action of the Jacobian on the vector is computed. $\left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}\right)^T$ is commonly referred to as the *adjoint operator*. By taking the transpose, we reverse the flow of information in the equations system. For example, if a tracer is advected downstream (and so information about upstream conditions is advected with it), the adjoint PDE advects information in the reverse sense, i.e. upstream. In fact, the adjoint system is the same idea as the reverse mode of algorithmic or automatic differentiation. Each component of the solution will have a corresponding adjoint variable. For example, if $\boldsymbol{F}$ is the Navier-Stokes equations, and $\boldsymbol{u}$ is the tuple of velocity and pressure, then $\boldsymbol{\lambda}$ is the tuple of adjoint velocity and adjoint pressure. The adjoint variable can then be used to compute the gradient of the loss function w.r.t. the parameters $\boldsymbol{\omega}$ as

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\omega}} = -\boldsymbol{\lambda}^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\varphi}} \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{\omega}}, \tag{2.56}$$

where the last of the gradient matrix is obtained by the neural network auto-differentiation. Usually, solving the adjoint system is extremely efficient when there are a small number of functionals (outputs), and a large number of parameters (inputs), as the adjoint method is generally independent of the number of parameters. This situation is very common PDE constrained optimization, where there is usually one functional (output), but many parameters. For further details, the reader is directed to [174] and to Paper A [47], where we employ the adjoint method to make the RANS solver differentiable and use it to solve the closure problem.

3 Summary of Publications

This chapter provides short summaries of the publications that are part of this thesis. The summaries are intended to provide a brief overview of the research problem, methodology, and the main results of the publication. The summaries are organized in the order in which the publications are attached in full in the thesis. The publications are attached in full in respective appendix chapters in Part III.

3.1 Paper A

A probabilistic, data-driven closure model for RANS simulations with aleatoric, model uncertainty
A. Agrawal, P.S. Koutsourelakis

Summary

The paper presents a novel, probabilistic, data-driven closure model for Reynolds-Averaged Navier-Stokes (RANS) simulations. The model leverages a hybrid approach that tightly integrates a RANS solver with a machine learning (ML) model, enabling end-to-end gradient-based learning through an adjoint-based differentiable solver. The ML model is designed with hard constraints to ensure physical symmetry and invariance, and it employs Bayesian learning using sparse, indirect data such as mean velocity and pressure, rather than direct Reynolds stress data. The closure model incorporates both a parametric component, using neural-network-based tensor basis functions, and a stochastic component that accounts for aleatoric model uncertainties. A fully Bayesian formulation is proposed, combined with a sparsity-inducing prior in order to identify regions in the problem domain where the parametric closure is insufficient, allowing for stochastic corrections to the Reynolds stress tensor. The proposed approach uses Stochastic Variational Inference with Monte Carlo estimates and the reparameterization trick, combining the parametric sensitivities from the differentiable RANS solver with automatic differentiation from the neural network. The model is shown to produce accurate, probabilistic predictions even in regions with significant model errors, as demonstrated in the complex case of flow recirculation and separation, the backward-facing step benchmark problem. In most cases, very good agreement with the reference values were achieved and in all cases these were enveloped by the credible intervals computed.

Contribution

(AA): Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **(PSK)**: Supervision, Writing – review & editing.

Reference

Atul Agrawal and Phaedon-Stelios Koutsourelakis. A probabilistic, data-driven closure model for RANS simulations with aleatoric, model uncertainty. *Journal of Computational Physics*, page 112982, 2024

3.2 Paper B

Physics-Informed Tensor Basis Neural Network for Turbulence Closure Modeling
L. Riccius, A. Agrawal, P.S. Koutsourelakis

Summary

This work introduces Physics Informed Tensor Basis Neural Network (PI-TBNN), which extends the TBNN framework with an extensive feature set and an inductive bias in the form of a physics-informed addition to the loss function. The addition to the loss function acts as a soft constraint, designed to enhance predictions of the anisotropy tensor in turbulent flow simulations, thereby improving the model's ability to predict complex turbulence behaviors, which are often beyond the capability of traditional Linear Eddy Viscosity Models (LEVMs). The model is further refined by enforcing physical constraints such as symmetry and invariance as hard constraints (already proposed in the TBNN framework), ensuring consistency with fundamental fluid dynamics principles. Learning is performed using sparse observations of the Reynolds stress tensor from various flow configurations, allowing the model to generalize effectively even in challenging scenarios. The proposed PI-TBNN method demonstrates significant improvements over the TBNN in predicting anisotropy tensors in unseen flow configurations, particularly in cases involving surface curvature and flow separation, which are traditionally difficult for LEVMs to handle. These advancements are visually confirmed using barycentric map visualizations and quantified by comparing relevant error matrices.

Contribution

(LR): Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **(AA)**: Conceptualization, Methodology, Writing – review & editing. **(PSK)**: Supervision, Writing – review & editing.

Reference

Leon Riccius, Atul Agrawal, and Phaedon-Stelios Koutsourelakis. Physics-informed tensor basis neural network for turbulence closure modeling. *Workshop on Machine Learning and the Physical Sciences (NeurIPS 2023)*, 2023

3.3 Paper C

From concrete mixture to structural design - a holistic optimization procedure in the presence of uncertainties

A. Agrawal, E. Tamsen, J. F. Unger, P.S. Koutsourelakis

Summary

This work introduces a systematic design framework aimed at promoting sustainability within the precast concrete industry. By employing a holistic optimization procedure, the framework seamlessly integrates concrete mixture design with structural simulations in a joint, forward workflow that is ultimately designed to be inverted. This approach allows for the exploration of new concrete mixtures beyond traditional ranges, ensuring that the design process accommodates various uncertainties, whether aleatoric or epistemic, particularly when calibrating physical models or identifying gaps in the workflow. One of the central challenges in this process is inverting the established causal relationships, especially when dealing with physics-based models that often lack derivative or sensitivity information, or when design constraints are present. To address these challenges, the work advocates for the use of Variational Optimization, enhanced with specific extensions and heuristics, to overcome the difficulties of handling black-box solvers and non-linear constraints within the workflow, while also managing parametric uncertainties. The framework also includes a model discovery method that leverages noisy and incomplete experimental data, along with finite element simulations, to learn the missing links between concrete mixture design variables and the parameters used in physics-based models. The applicability of this comprehensive optimization framework is demonstrated through the design of a precast concrete beam. The objective of this design is to minimize the Global Warming Potential while adhering to performance constraints such as load-bearing capacity, demolding time, and maximum temperature during hydration, all in accordance with Eurocode standards. This approach is not only relevant to the precast concrete industry but is also transferable to various other materials, structural, and mechanical design problems.

Contribution

(AA): (Machine learning aspects) Methodology, Software, Visualization, Writing – review & editing **(E.T.)**: (material science application aspects) Methodology, Software, Visualization, Writing – review & editing. **(J.F.U.)**: Supervision, Writing – review & editing, code review **(PSK)**: Supervision, Writing – review & editing

Reference

Atul Agrawal, Erik Tamsen, Phaedon-Stelios Koutsourelakis, and Joerg F Unger. From concrete mixture to structural design–a holistic optimization procedure in the presence of uncertainties. *Data-Centric Engineering*, pages 1–32, 2024

3.4 Paper D

Multi-fidelity Constrained Optimization for Stochastic Black-Box Simulators
A. Agrawal, K. Ravi, P.S. Koutsourelakis, H.J. Bungartz

Summary

This work presents a novel algorithm, Scout-Nd (Stochastic Constrained Optimization for N dimensions), designed for stochastic constraint optimization where both objectives and constraints involve black-box physics-based models. The algorithm addresses the challenges inherent in optimization problems involving physics-based simulators that are stochastic, computationally expensive, multi-modal and operate in high-dimensional parameter spaces. Traditional optimization methods often rely on gradient information, which is typically unavailable in such legacy black-box codes. To overcome the difficulty of unavailable gradients in legacy black-box codes, the algorithm utilizes efficient gradient estimation techniques that incorporate variance reduction strategies to improve accuracy and stability. Additionally, to balance the trade-off between computational cost and accuracy, the method integrates multi-fidelity strategies, reducing the need for multiple evaluations of expensive physics-based models. The proposed method demonstrates superior performance compared to standard black-box optimization methods, showcasing enhanced robustness and quality of the optimized solutions. The effectiveness of Scout-Nd is validated through standard benchmarks, where it consistently outperforms existing techniques, highlighting its potential for robust and efficient parameter optimization in complex design processes.

Contribution

(AA): Methodology, Software, Visualization, Writing – review & editing **(K.R.)**: Methodology, Software, Visualization, Writing – review & editing. **(P.S.K)**: Methodology, Supervision, Writing – review & editing **(HJB)**: Supervision, Writing – review.

Reference

Atul Agrawal, Kislaya Ravi, Phaedon-Stelios Koutsourelakis, and Hans-Joachim Bungartz. Multi-fidelity constrained optimization for stochastic black-box simulators. *Workshop on Machine Learning and the Physical Sciences (NeurIPS 2023)*, 2023

3.5 Paper E

Stochastic Black-Box Simulator Optimization using Multi-Fidelity Score Function Estimator

A. Agrawal, K. Ravi, P.S. Koutsourelakis, H.J. Bungartz

Summary

The paper extends our previously proposed SCOUT-Nd algorithm by incorporating several key enhancements. By utilizing natural gradients in the gradient estimation process, the algorithm achieves well-behaved convergence properties and produces a higher quality of optima, demonstrated through numerical examples. To efficiently manage multi-modal optimization landscapes, smart heuristics are employed, which have been validated through a series of academic examples. The study provides a detailed mathematical analysis of the method's convergence, ensuring the robustness of the approach. Additionally, an adaptive sample size mechanism for the gradient estimation is implemented, significantly reducing the number of solver calls required to reach the optimum in numerical tests. The adaptive sample size is also adapted to the multi-fidelity framework of the Scout-Nd algorithm. The algorithm's capabilities are further tested against an expanded set of academic examples, comparing its performance on challenges such as multi-modality, constraints, valley behavior, and dimension scalability. In all these tests, the algorithm demonstrates superior performance in parameter optimization compared to existing black-box optimization methods. Additionally, we showcase the algorithm's efficacy in a complex real-world application: optimizing wind farm layout. It outperforms standard baseline methods in both robustness and the quality of the achieved optimum. Also, in this case the multi-fidelity framework of SCOUT-Nd produced comparable accuracy at a one-third cost.

Contribution

(AA): Methodology, Software, Visualization, Writing – review & editing **(K.R.)**: Methodology, Software, Visualization, Writing – review & editing. **(P.S.K)**: Methodology, Supervision, Writing – review & editing **(HJB)**: Supervision, Writing – review.

Reference

Atul Agrawal, Kislaya Ravi, Phaedon-Stelios Koutsourelakis, and Hans-Joachim Bungartz. Stochastic black-box optimization using multi-fidelity score function estimator. *Machine Learning: Science and Technology*, 6(1):015024, jan 2025

PART II

CONCLUSION

Pippin: "I didn't think it would end this way."
Gandalf: "End? No, the journey doesn't end here.[...]"

The Lord of the Rings: The Return of the King
J.R.R. Tolkien [176]

4 Summary and conclusion

In this chapter, we recap the motivation behind the thesis, and the contributions made, and we point out the major outstanding challenges and the future work that can be done to address these challenges. The thesis is motivated by the promise ML brings to advance scientific and engineering disciplines. However, the quest to combine ML with scientific models is crippled because of several pitfalls. To point out a few, the lack of large labeled datasets in scientific applications, datasets spanning only a narrow segment of possible data distributions leading to poor generalization abilities, predictions being scientifically invalid, presence of noise in data, and difficulty coupling ML models with scientific models because of lack of differentiability of the physics-based models.

In this thesis, we aim to address the above challenges by proposing novel strategies that combine machine learning with scientific understanding (e.g., governing equations, symmetries, invariances etc.) to develop scientific machine learning models aided by the differentiable physics, which are then tightly coupled with probabilistic approaches. These strategies are used in addressing challenging computational physics and engineering downstream tasks like inference, uncertainty quantification, and optimization for complex applications like turbulence closure modeling, concrete material and structural design, and windfarm layout optimization. Data from physical systems can be sparse and expensive to obtain. Numerical simulations are computationally costly as the dimensions involved are, in general, very high, whereas the time and spatial scales of interest are very small. Experiments are equally expensive and can be infeasible given the spatial scales of the system. Adding domain scientific knowledge i.e., governing equations and underlying physical principles in the form of inductive bias supplements the need for extensive labeled data, thus making the learning data parsimonious. Furthermore, adding physical knowledge to the ML models improves the generalization abilities and makes the predictions scientifically valid. The differentiability of physics-based models facilitates the integration of ML models with scientific models, allowing for inference and optimization in high-dimensional parameter spaces that would otherwise be computationally infeasible. However, while physics-based inductive bias can improve ML models, it can also be problematic if the bias is incorrect, potentially leading to inaccurate predictions. To mitigate this risk, we adopt probabilistic approaches, specifically the Bayesian approach, in this thesis. The probabilistic approach helps in accounting for different types of uncertainties arising due to noisy data, model misspecification, and the inherent stochasticity in the physical systems. In scientific and engineering applications, quantifying uncertainty in predictions is crucial for informed decision-making, and probabilistic approaches are instrumental in achieving this goal. The rest of the chapter is organized as follows: Section 4.1 summarizes the scientific achievements of the author and Section 4.2 discusses the outstanding challenges and future work.

4.1 What was achieved

The author's scientific achievements are presented in the form of a) publications summarized in Chapter 3, given in full in the Part III and categorized in Table 1.1, and b) broad theoretical background with an up-to-date literature survey of the dynamic field of SciML and probabilistic modeling presented in Chapter 2.

In this thesis, we started by approaching turbulence in fluid mechanics which as famously quoted by Feynman [177] is "the most important unsolved problem of classical physics". The study of fluid turbulence gave birth to several fields in computational mathematics [23, 24]. In this thesis, we drew ideas from SciML, differentiable physics and probabilistic modeling to propose novel strategies to address the turbulence closure problem (Paper A [47] and Paper B [48]). We later extended the broad ideas to material and structural design optimization (Paper C [49]). The later investigation led us to study optimization under uncertainty for black box physics-based models (Paper D [50] and Paper E [51]).

In Paper A [47] and Paper B [48], we approached the turbulence closure problem in fluid mechanics employing the aforementioned strategies. In Paper A [47] we addressed all the challenges discussed in Section 1.1 in the context of turbulence closure modeling. In this context, the challenges have a significant overlap. In particular, we proposed a probabilistic, data-driven closure model for Reynolds-Averaged Navier-Stokes (RANS) simulations. Learning is conducted in a Bayesian framework, utilizing sparse indirect high-fidelity data of mean velocity and pressure as the training data. The model follows a hybrid approach of Scientific Machine Learning (SciML), combining the RANS solver and the machine learning model in a tightly coupled manner. To facilitate end-to-end gradient-based learning, an adjoint-based differentiable RANS solver is developed. The proposed closure consists of two parts. First, neural-network-based tensor basis functions that are dependent on the rate of strain and rotation tensor invariants. This is complemented by latent, random variables which account for aleatoric model errors. The machine learning model is designed to incorporate physical constraints, such as symmetry and invariance, as hard constraints. Sparsity-inducing prior is used in order to identify the regions in the problem domain where the parametric closure is insufficient and in order to quantify the stochastic correction to the Reynolds stress tensor. We demonstrated the computation of probabilistic predictive estimates for all output quantities of the trained RANS model, illustrating their accuracy on a separated flow in the backward-facing step benchmark problem. The results showed very good agreement with reference values, with all cases falling within the computed credible intervals. The Paper B [48] addressed challenges (i) and (ii) (ref. Section 1.1), extends the work of [28] by incorporating violation of realizability constraints of the anisotropic part of Reynolds stress tensor as a hard constraint. Here the learning is performed with sparse observations of the Reynolds stress tensor. We demonstrated that the proposed enhancements improve anisotropy tensor predictions in unseen challenging cases involving surface curvature and flow separation.

In Paper C [49], we addressed challenges (ii), (iii), (iv) and (v) (ref. Section 1.1). We presented an optimization framework that integrates physics-based models, empirical relations, and experimental data, which are typically used separately in the design of concrete structural systems. This framework combines concrete mixture design and structural simulation, utilizing semi-analytical models, finite-element solvers, and machine learning models. Additionally, we introduced a model discovery method using the Variational Bayes Expectation-maximization scheme to learn the missing links between concrete mixture design variables and parameters in physics-based models using noisy and incomplete experimental data. Our optimization framework is capable of handling black-box solvers, non-linear constraints, and parametric uncertainties. Finally, we demonstrated the application of this framework in the mixture design of a concrete beam, minimizing the global warming potential while satisfying various performance constraints. The framework presented is demonstrated on the specific design problem, but it is general and can be applied to other design problems involving complex material systems.

Motivated by the challenging problem of (constrained) optimization involving black-box, stochastic, high-dimensional, and expensive-to-evaluate physics-based models, we proposed a novel optimization algorithm in Paper D [50] and extended it in Paper E [51]. Both of the papers are aimed at addressing challenges (iv) and (v) (ref. Section 1.1). The proposed algorithm relies on efficient gradient estimation using score function estimates with suitable variance reduction strategies. To balance computational cost and accuracy, the algorithm incorporates several state-of-the-art techniques. These include multi-fidelity methods for gradient estimation, natural gradients to enhance convergence properties and optimum quality, Quasi-Monte Carlo (QMC) methods for gradient estimation, and a novel adaptive Monte Carlo sample-size selection for the gradient estimation. The algorithm is tested against several standard baseline algorithms on a suite of benchmark problems involving different types of optimization challenges like multi- modality, constraints, behavior in valleys, dimension scalability, etc. The results show that the proposed algorithm outperforms the state-of-the-art optimization algorithms. The proposed algorithm could also handle very complex multi-modal surfaces and high-dimensional optimization problems. The algorithm is tested on a real-world complex case of wind farm layout optimization problem, where the objective is to maximize the expected energy production of the wind farm while satisfying the constraints on the turbine locations. The results show that the proposed algorithm outperforms the state-of-the-art optimization algorithms in terms of robustness and quality of the optimum.

4.2 Outstanding challenges and future work

We discuss the challenges and the potential future direction to address those challenges in detail in the respective publication. Here, we provide a brief overview from a broader and overarching perspective.

- **Spatial awareness.** The neural network based closure model in Paper A [47] and Paper B [48] follows a locality assumption, which is a very strong assumption for flows that exhibit strong inhomogeneity. Non-local features of the velocity field might be needed to better inform the model. This could be achieved by employing models that embed a certain sense of spatial awareness, e.g., the Convolutional Neural Networks (CNN). In the context of turbulence closure modeling, vector-cloud neural networks (VCNN) [178] with the appropriate embedding of invariance properties seems promising.

- **Challenges posed by differentiability.** The end-to-end training in Paper A [47] hinges on the availability of a differentiable RANS solver. Also, the optimization of the concrete beam design in Paper C [49] relies on the gradient information from the material and structural simulation joint workflow. Rewriting a complex solver such as the RANS in a differentiable programming language like JAX or PyTorch would be non-trivial endeavor. The challenge of obtaining gradients in a more general setting, where the solver is not differentiable, remains open for many legacy solvers. There exist solvers written using differentiable programming languages, but these solvers are often written for a specific application/use case, limited in their capabilities and are not as accurate as legacy solvers. There is a growing need for differentiable solvers that are accurate, general, and can be used across various applications, with [38] taking the initial steps towards this goal. So we resorted to obtaining the gradients by the adjoint method in Paper A [47], which required appropriate adjustments in both the solver and the inference algorithm. The challenge is further compounded when the solver is to be coupled with a probabilistic programming framework like Pyro [89]. These infrastructural and implementational hurdles pose a major challenge for researchers to adopt the end-to-end training paradigm in a probabilistic setting. Recently, automatic differentiable (AD) compiler methods [171] have emerged, providing gradients of legacy codes written in C, C++, Fortran, etc. These methods could potentially be used to obtain gradients of legacy solvers addressed in the present thesis, which could then be coupled with e.g., probabilistic programming frameworks.

- **Interpretability and generalization.** The machine learning closure models discussed in Paper A [47] and Paper B [48] are generally represented by neural networks that are limited in their sense of interpretability. However, interpretability is a key requirement for machine-learned closure models to find their way into engineering practice. Consequently, recent research efforts [179, 180] are focused on developing interpretable, closed-form expressions for the closure model. For example, techniques such as sparse linear regression with a physics-informed library of candidate functions are being employed. Beyond interpretability, generalizability is a crucial research area essential for the widespread adoption of trained closure models across various scenarios. Developing a model trained on specific parameters (e.g., Reynolds number), geometries, boundary conditions, and initial conditions, and then successfully applying it beyond this parameter set, is often regarded as the *holy grail* of closure modeling research. However, achieving this level of generality remains an open problem [4, 52].

- **Network architectures.** Choosing the right network architecture and optimizing its parameters is often more of an art than a science. In this thesis, an extensive study has not been performed on the choice of different network architectures used in the Paper A [47], Paper B [48] and Paper C [49] because new architectures are appearing continuously and because it is still highly non-trivial which one to choose. To elaborate further, the neural network architectures used in Paper A [47] and Paper B [48] are relatively simple, consisting of a few fully connected layers. More complex architectures, such as graph neural networks (GNNs) or transformer networks, could potentially capture the underlying physics more effectively. For example, GNNs and CNNs have been shown to be effective in learning the structure of the data and could be used to learn the spatial relationships in the flow field [151]. Similarly, transformer networks have been shown to be effective in capturing long-range dependencies in the data and could be used to capture the interactions between different parts of the flow field [17]. Exploring these more complex architectures could potentially lead to more accurate and generalizable closure models.

- **Benchmark datasets.** The quality of the models learned in Paper A [47] and Paper B [48] are reliant on the quality of the training data. Unfortunately, the field of flow physics lacks a suitable large dataset such as ImageNet for photos [11]. The reason for this lack is that the scientific data is generally high-dimensional. To set this into perspective, GPT-3 was trained with six hundred gigabytes of text data. In contrast, BLASTNet [181], a recent dataset for fluid flows and the largest of its kind is five terabytes. Prior to BLASTNet, the largest dataset was the Johns Hopkins Turbulence Database [182] which is limited in flow configurations provided thus limiting the generalization abilities of the models trained on it. The lack of benchmark datasets in scientific applications is a major bottleneck for the development of machine learning models. Efforts towards creating benchmark datasets for scientific applications are necessary to advance the field of scientific machine learning.

- **Dimensionality.** The dimensionality reduction of the stochastic discrepancy terms in Paper A [47] utilized a pre-determined and uniform division of the problem domain into subdomains. The model's accuracy could be significantly improved with a learnable and adaptive scheme that focuses on areas with the most pronounced model deficiencies, where stochastic corrections are most necessary. This would require a more sophisticated treatment of the sparsity inducing prior.

- **Extension to other applications.** The approach presented in Paper C [49] is transferable to several other materials, structural and mechanical problems. Such extensions could readily include more complex design processes with an increased number of parameters and constraints (the latter due to multiple load configurations or limit states in a real structure). Furthermore, this procedure could be applied to problems involving a complete structure (e.g., bridge, building) instead of a single element and potentially entailing advanced modeling features that include multiscale models to link material composition to material properties or improve the computation of the global warming potential using a complete life cycle analysis. The black-box optimization approach presented in Paper D [50] and Paper E [51] can be tested on other challenging real-world applications within the scope of the proposed method, such as particle physics [138]. However, if the physics-based models are prohibitively expensive, the gradient estimation can benefit from further variance reduction, potentially achievable through importance sampling.

This thesis bridges some of the gaps and tackles some of the challenges in the broad field of SciML. However, the field of SciML is rapidly evolving, and numerous research questions extend beyond the scope of this thesis. These questions are crucial for future exploration. For instance, will the most effective SciML approaches offer general techniques applicable across various domains, or will they remain highly domain-specific? What is the optimal balance between hard-coded physical principles and learned components in designing SciML algorithms? Can SciML contribute new perspectives and ideas to the field of ML? Answering these questions requires a truly interdisciplinary effort. Consequently, SciML is an exciting and fast-growing field, with many discoveries still to come, and its impact on various domains is likely to grow as the field continues to develop. Similar to the role played by internet-scale data in computer vision and text translation problems in revolutionizing the field of deep learning research, we anticipate the richness of challenges and opportunities encountered in SciML will influence the next major revolution in ML research.

Bibliography

[1] Ivan Marusic and Susan Broomhall. Leonardo da vinci and fluid mechanics. *Annual Review of Fluid Mechanics*, 53(1):1–25, 2021.

[2] Stephen B Pope. *Turbulent flows*. Cambridge university press, 2000.

[3] Muhammad Kasim, Duncan Watson-Parris, Lucia Deaconu, Sophy Oliver, Peter Hatfield, Dustin Froula, Gianluca Gregori, Matt Jarvis, Samar Khatiwala, Jun Korenaga, et al. Up to two billion times acceleration of scientific simulations with deep neural architecture search. In *APS division of plasma physics meeting abstracts*, volume 2020, pages BO05–001, 2020.

[4] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51:357–377, 2019.

[5] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

[6] Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

[7] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating scientific knowledge with machine learning for earth and environmental systems: A survey. *ACM Computing Survey*, 2022.

[8] Phaedon-Stelios Koutsourelakis, Nicholas Zabaras, and Michele Girolami. Big data and predictive computational modeling. *Journal of Computational Physics*, 321:1252–1254, 2016.

[9] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[12] Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, Yaochen Xie, Meng Liu, Yuchao Lin, Zhao Xu, Keqiang Yan, Keir Adams, Maurice Weiler, Xiner Li, Tianfan Fu, Yucheng Wang, Haiyang Yu, YuQing Xie, Xiang Fu, Alex Strasser, Shenglong Xu, Yi Liu, Yuanqi Du, Alexandra Saxton, Hongyi Ling, Hannah Lawrence, Hannes Stärk, Shurui Gui, Carl Edwards, Nicholas Gao, Adriana Ladera, Tailin Wu, Elyssa F. Hofgard, Aria Mansouri Tehrani, Rui Wang, Ameya Daigavane, Montgomery Bohde, Jerry Kurtin,

Qian Huang, Tuong Phung, Minkai Xu, Chaitanya K. Joshi, Simon V. Mathis, Kamyar Azizzadenesheli, Ada Fang, Alán Aspuru-Guzik, Erik Bekkers, Michael Bronstein, Marinka Zitnik, Anima Anandkumar, Stefano Ermon, Pietro Liò, Rose Yu, Stephan Günnemann, Jure Leskovec, Heng Ji, Jimeng Sun, Regina Barzilay, Tommi Jaakkola, Connor W. Coley, Xiaoning Qian, Xiaofeng Qian, Tess Smidt, and Shuiwang Ji. Artificial Intelligence for Science in Quantum, Atomistic, and Continuum Systems, November 2023. arXiv:2307.08423 [physics].

[13] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[14] Anuj Karpatne, Ramakrishnan Kannan, and Vipin Kumar. *Knowledge Guided Machine Learning: Accelerating Discovery Using Scientific Knowledge and Data.* CRC Press, 2022.

[15] Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K Gupta, and Aditya Grover. Climax: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343*, 2023.

[16] Maike Sonnewald, Redouane Lguensat, Daniel C Jones, Peter D Dueben, Julien Brajard, and Venkatramani Balaji. Bridging observations, theory and numerical simulation of the ocean using machine learning. *Environmental Research Letters*, 16(7):073008, 2021.

[17] Cristian Bodnar, Wessel P Bruinsma, Ana Lucic, Megan Stanley, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan Weyn, Haiyu Dong, Anna Vaughan, et al. Aurora: A foundation model of the atmosphere. *arXiv preprint arXiv:2405.13063*, 2024.

[18] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[19] Ricardo Vinuesa, Jean Rabault, Hossein Azizpour, Stefan Bauer, Bingni W. Brunton, Arne Elofsson, Elias Jarlebring, Hedvig Kjellstrom, Stefano Markidis, David Marlevi, Paola Cinnella, and Steven L. Brunton. Opportunities for machine learning in scientific discovery, May 2024. arXiv:2405.04161 [cs].

[20] Heng Xiao and Paola Cinnella. Quantification of model uncertainty in RANS simulations: A review. *Progress in Aerospace Sciences*, 108:1–31, 2019.

[21] Yanay Rosen, Yusuf Roohani, Ayush Agrawal, Leon Samotorcan, Tabula Sapiens Consortium, Stephen R Quake, and Jure Leskovec. Universal cell embeddings: A foundation model for cell biology. *bioRxiv*, pages 2023–11, 2023.

[22] Tommaso Dorigo, Andrea Giammanco, Pietro Vischia, Max Aehle, Mateusz Bawaj, Alexey Boldyrev, Pablo de Castro Manzano, Denis Derkach, Julien Donini, Auralee Edelen, Federica Fanzago, Nicolas R. Gauger, Christian Glaser, Atılım G. Baydin, Lukas Heinrich, Ralf Keidel, Jan Kieseler, Claudius Krause, Maxime Lagrange, Max Lamparth, Lukas Layer, Gernot Maier, Federico Nardi, Helge E. S. Pettersen, Alberto Ramos, Fedor Ratnikov, Dieter Röhrich, Roberto Ruiz de Austri, Pablo Martínez Ruiz del Árbol, Oleg Savchenko, Nathan Simpson, Giles C. Strong, Angela Taliercio, Mia Tosi, Andrey Ustyuzhanin, and Haitham Zaraket. Toward the end-to-end optimization of particle physics instruments with differentiable programming. *Reviews in Physics*, 10:100085, June 2023.

[23] Charles L Fefferman. Existence and smoothness of the Navier-Stokes equation. *The millennium prize problems*, 57:67, 2000.

[24] Shady E. Ahmed, Suraj Pawar, Omer San, Adil Rasheed, Traian Iliescu, and Bernd R. Noack. On closures for reduced order models - A spectrum of first-principle to machine-learned avenues. *Physics of Fluids*, 33(9):091301, September 2021. arXiv: 2106.14954.

[25] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[26] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[27] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[28] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

[29] Nicholas Geneva and Nicholas Zabaras. Quantifying model form uncertainty in Reynolds-averaged turbulence models with bayesian deep neural networks. *Journal of Computational Physics*, 383:125–147, 2019.

[30] Mikael LA Kaandorp and Richard P Dwight. Data-driven modelling of the Reynolds stress tensor using random forests with invariance. *Computers & Fluids*, 202:104497, 2020.

[31] Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Griffin Mooers, Milan Klöwer, James Lottes, Stephan Rasp, Peter Düben, et al. Neural general circulation models for weather and climate. *Nature*, pages 1–7, 2024.

[32] Edwin T Jaynes. *Probability theory: The logic of science.* Cambridge university press, 2003.

[33] William Briggs. *Uncertainty: the soul of modeling, probability & statistics.* Springer, 2016.

[34] Phaedon-Stelios Koutsourelakis. Accurate uncertainty quantification using inaccurate computational models. *SIAM Journal on Scientific Computing*, 31(5):3274–3300, 2009.

[35] Phaedon-Stelios Koutsourelakis. Stochastic upscaling in solid mechanics: An excercise in machine learning. *Journal of Computational Physics*, 226(1):301–325, 2007.

[36] Mathieu Blondel and Vincent Roulet. The Elements of Differentiable Programming, March 2024. arXiv:2403.14606 [cs].

[37] Charles Audet and Michael Kokkolaras. Blackbox and derivative-free optimization: theory, algorithms and applications, 2016.

[38] Philipp Holl and Nils Thuerey. ϕ-ML: Intuitive scientific computing with dimension types for jax, pytorch, tensorflow numpy. *Journal of Open Source Software*, 9(95):6171, 2024.

[39] Björn List, Li-Wei Chen, and Nils Thuerey. Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons. *Journal of Fluid Mechanics*, 949:A25, 2022.

[40] Jonas Degrave, Michiel Hermans, Joni Dambre, et al. A differentiable physics engine for deep learning in robotics. *Frontiers in neurorobotics*, page 6, 2019.

[41] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. *Advances in Neural Information Processing Systems*, 31, 2018.

[42] Carlos A Michelén Ströfer and Heng Xiao. End-to-end differentiable learning of turbulence models from indirect observations. *Theoretical and Applied Mechanics Letters*, 11(4):100280, 2021.

[43] Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. *Advances in Neural Information Processing Systems*, 33:6111–6122, 2020.

[44] Oliver Brenner, Pasha Piroozmand, and Patrick Jenny. Efficient assimilation of sparse data into RANS-based turbulent flow simulations using a discrete adjoint method. *Journal of Computational Physics*, 471:111667, 2022.

[45] Michael B Giles and Niles A Pierce. An introduction to the adjoint approach to design. *Flow, turbulence and combustion*, 65(3):393–415, 2000.

[46] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[47] Atul Agrawal and Phaedon-Stelios Koutsourelakis. A probabilistic, data-driven closure model for RANS simulations with aleatoric, model uncertainty. *Journal of Computational Physics*, page 112982, 2024.

[48] Leon Riccius, Atul Agrawal, and Phaedon-Stelios Koutsourelakis. Physics-informed tensor basis neural network for turbulence closure modeling. *Workshop on Machine Learning and the Physical Sciences (NeurIPS 2023)*, 2023.

[49] Atul Agrawal, Erik Tamsen, Phaedon-Stelios Koutsourelakis, and Joerg F Unger. From concrete mixture to structural design–a holistic optimization procedure in the presence of uncertainties. *Data-Centric Engineering*, pages 1–32, 2024.

[50] Atul Agrawal, Kislaya Ravi, Phaedon-Stelios Koutsourelakis, and Hans-Joachim Bungartz. Multi-fidelity constrained optimization for stochastic black-box simulators. *Workshop on Machine Learning and the Physical Sciences (NeurIPS 2023)*, 2023.

[51] Atul Agrawal, Kislaya Ravi, Phaedon-Stelios Koutsourelakis, and Hans-Joachim Bungartz. Stochastic black-box optimization using multi-fidelity score function estimator. *Machine Learning: Science and Technology*, 6(1):015024, jan 2025.

[52] Benjamin Sanderse, Panos Stinis, Romit Maulik, and Shady E. Ahmed. Scientific machine learning for closure models in multiscale problems: a review, March 2024. arXiv:2403.02913 [cs, math].

[53] David Corfield and Jon Williamson. *Foundations of Bayesianism*, volume 24. Springer Science & Business Media, 2013.

[54] Tim Sullivan. *Introduction to Uncertainty Quantification.* Texts in Applied Mathematics, 63, Band 63. Springer, 2015.

[55] Paul Messina, David Brown, et al. Scientific grand challenges in national security: The role of computing at the extreme scale. In *Workshop Report, Department of Energy, October*, 2009.

[56] Ralph C Smith. *Uncertainty quantification: theory, implementation, and applications.* SIAM, 2013.

[57] David JC MacKay. *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

[58] Sheldon M Ross. *Introduction to probability models.* Academic press, 2014.

[59] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[60] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.

[61] Kevin P Murphy. *Probabilistic machine learning: Advanced topics.* MIT press, 2023.

[62] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

[63] Olivier Le Maître and Omar M Knio. *Spectral methods for uncertainty quantification: with applications to computational fluid dynamics.* Springer Science & Business Media, 2010.

[64] Norbert Wiener. The homogeneous chaos. *American Journal of Mathematics*, 60(4):897–936, 1938.

[65] George Fishman. *Monte Carlo: concepts, algorithms, and applications.* Springer Science & Business Media, 2013.

[66] Russel E Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta numerica*, 7:1–49, 1998.

[67] Søren Asmussen and Peter W Glynn. *Stochastic simulation: algorithms and analysis*, volume 57. Springer, 2007.

[68] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *Siam Review*, 60(3):550–591, 2018.

[69] Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of Monte Carlo methods.* John Wiley & Sons, 2013.

[70] Jacques Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin*, pages 49–52, 1902.

[71] Kazufumi Ito and Bangti Jin. *Inverse problems: Tikhonov theory and algorithms*, volume 22. World Scientific, 2014.

[72] Jonas Latz. On the well-posedness of bayesian inverse problems. *SIAM/ASA Journal on Uncertainty Quantification*, 8(1):451–482, 2020.

[73] David JC MacKay. Probable networks and plausible predictions-a review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469, 1995.

[74] Johnathan M Bardsley. Gaussian markov random field priors for inverse problems. *Inverse Problems & Imaging*, 7(2):397, 2013.

[75] Michael Riis Andersen, Aki Vehtari, Ole Winther, and Lars Kai Hansen. Bayesian inference for spatio-temporal spike-and-slab priors. *The Journal of Machine Learning Research*, 18(1):5076–5133, 2017.

[76] Anqi Wu, Mijung Park, Oluwasanmi O Koyejo, and Jonathan W Pillow. Sparse bayesian structure learning with "dependent relevance determination" priors. In *Advances in Neural Information Processing Systems*, pages 1628–1636, 2014.

[77] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):611–622, 1999.

[78] M. E. Tipping. The Relevance Vector Machine. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 652–658. MIT Press, 2000.

[79] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International conference on machine learning*, pages 324–333. PMLR, 2016.

[80] Mike West and Michael D Escobar. *Hierarchical priors and mixture models, with application in regression and density estimation.* Institute of Statistics and Decision Sciences, Duke University, 1993.

[81] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques.* MIT press, 2009.

[82] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.

[83] Michael Irwin Jordan. *Learning in graphical models.* MIT press, 1999.

[84] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[85] Johan Ludwig William Valdemar Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30(1):175–193, 1906.

[86] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[87] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[88] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[89] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

[90] Dustin Tran, Matthew D Hoffman, Rif A Saurous, Eugene Brevdo, Kevin Murphy, and David M Blei. Deep probabilistic programming. *arXiv preprint arXiv:1701.03757*, 2017.

[91] Matthew James Beal. *Variational algorithms for approximate Bayesian inference.* University of London, University College London (United Kingdom), 2003.

[92] Jorge Nocedal and Stephen J Wright. *Numerical optimization.* Springer, 1999.

[93] Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 71(2):319–392, 2009.

[94] Luke Tierney and Joseph B Kadane. Accurate approximations for posterior moments and marginal densities. *Journal of the american statistical association*, 81(393):82–86, 1986.

[95] Giorgio Parisi and Ramamurti Shankar. Statistical field theory. 1988.

[96] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[97] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[98] Martin Jankowiak and Fritz Obermeyer. Pathwise derivatives beyond the reparameterization trick. In *International conference on machine learning*, pages 2235–2244. PMLR, 2018.

[99] Tijmen Tieleman. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26, 2012.

[100] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial intelligence and statistics*, pages 814–822. PMLR, 2014.

[101] Abhinav Agrawal, Daniel R Sheldon, and Justin Domke. Advances in black-box VI: Normalizing flows, importance weighting, and optimization. *Advances in Neural Information Processing Systems*, 33:17358–17369, 2020.

[102] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[103] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte Carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244, 2020.

[104] Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

[105] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient Estimation Using Stochastic Computation Graphs, January 2016. arXiv:1506.05254 [cs].

[106] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.

[107] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

[108] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

[109] Radford M Neal and Geoffrey E Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, pages 355–368, 1998.

[110] Faming Liang, Chuanhai Liu, and Raymond Carroll. *Advanced Markov chain Monte Carlo methods: learning from past samples*. John Wiley & Sons, 2011.

[111] W Keith Hastings. Monte Carlo sampling methods using markov chains and their applications. 1970.

[112] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli*, pages 223–242, 2001.

[113] Luke Tierney. Markov chains for exploring posterior distributions. *the Annals of Statistics*, pages 1701–1728, 1994.

[114] Heikki Haario, Marko Laine, Antonietta Mira, and Eero Saksman. DRAM: efficient adaptive MCMC. *Statistics and computing*, 16:339–354, 2006.

[115] Radford M Neal. MCMC using hamiltonian dynamics. *arXiv preprint arXiv:1206.1901*, 2012.

[116] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[117] Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1):1–32, 2017.

[118] John Salvatier, Thomas V. Wiecki, and Christopher J. Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Prepr.*, 4:e1686, 2016.

[119] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential Monte Carlo methods. *Sequential Monte Carlo methods in practice*, pages 3–14, 2001.

[120] Christian A Naesseth, Fredrik Lindsten, Thomas B Schön, et al. Elements of sequential Monte Carlo. *Foundations and Trends® in Machine Learning*, 12(3):307–392, 2019.

[121] Jonathan Huggins, Mikolaj Kasprzak, Trevor Campbell, and Tamara Broderick. Validated variational inference via practical posterior error bounds. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1802. PMLR, 2020.

[122] Yuling Yao, Aki Vehtari, Daniel Simpson, and Andrew Gelman. Yes, but did it work?: Evaluating variational inference. In *International Conference on Machine Learning*, pages 5581–5590. PMLR, 2018.

[123] Roger B Grosse, Siddharth Ancha, and Daniel M Roy. Measuring the reliability of MCMC inference with bidirectional Monte Carlo. *Advances in Neural Information Processing Systems*, 29, 2016.

[124] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control.* Cambridge University Press, 2022.

[125] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

[126] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 2017.

[127] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[128] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[129] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[130] Didier Lucor, Atul Agrawal, and Anne Sergent. Simple computational strategies for more effective physics-informed neural networks modeling of turbulent natural convection. *Journal of Computational Physics*, 456:111022, 2022.

[131] Jonas Nitzler, Jonas Biehler, Niklas Fehn, Phaedon-Stelios Koutsourelakis, and Wolfgang A Wall. A generalized probabilistic learning approach for multi-fidelity uncertainty quantification in complex physical simulations. *Computer Methods in Applied Mechanics and Engineering*, 400:115600, 2022.

[132] Maximilian Rixner and Phaedon-Stelios Koutsourelakis. Self-supervised optimization of random material microstructures in the small-data regime. *npj Computational Materials*, 8(1):46, 2022.

[133] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 2021.

[134] Philipp Holl and Nils Thuerey. The Unreasonable Effectiveness of Solving Inverse Problems with Neural Networks, August 2024. arXiv:2408.08119 [cs].

[135] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[136] Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature communications*, 12(1):6136, 2021.

[137] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016. Publisher: National Acad Sciences.

[138] Sergey Shirobokov, Vladislav Belavin, Michael Kagan, Andrei Ustyuzhanin, and Atilim Gunes Baydin. Black-box optimization with local generative surrogates. *Advances in Neural Information Processing Systems*, 33:14650–14662, 2020.

[139] Clarence W Rowley, Tim Colonius, and Richard M Murray. Model reduction for compressible flows using pod and galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1-2):115–129, 2004.

[140] David J Lucia, Philip S Beran, and Walter A Silva. Reduced-order modeling: new approaches for computational physics. *Progress in aerospace sciences*, 40(1-2):51–117, 2004.

[141] Karthik Duraisamy. Perspectives on Machine Learning-augmented Reynolds-averaged and Large Eddy Simulation Models of Turbulence. *Physical Review Fluids*, 6(5):050504, May 2021. arXiv:2009.10675 [physics].

[142] Hannah Christensen and Laure Zanna. Parametrization in weather and climate models. 2022.

[143] Erik Van Der Giessen, Peter A Schultz, Nicolas Bertin, Vasily V Bulatov, Wei Cai, Gábor Csányi, Stephen M Foiles, Marc GD Geers, Carlos González, Markus Hütter, et al. Roadmap on multiscale materials modeling. *Modelling and Simulation in Materials Science and Engineering*, 28(4):043001, 2020.

[144] Salar Taghizadeh, Freddie D Witherden, and Sharath S Girimaji. Turbulence closure modeling with data-driven techniques: physical compatibility and consistency considerations. *New Journal of Physics*, 22(9):093023, 2020.

[145] Julia Ling, Reese Jones, and Jeremy Templeton. Machine learning strategies for systems with invariance properties. *Journal of Computational Physics*, 318:22–35, 2016. Publisher: Elsevier Inc.

[146] Heng Xiao, J-L Wu, J-X Wang, Rui Sun, and C Roy. Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier-Stokes simulations: A data-driven, physics-informed Bayesian approach. *Journal of Computational Physics*, 324:115–136, 2016.

[147] Stephen B Pope. A more general effective-viscosity hypothesis. *Journal of Fluid Mechanics*, 72(2):331–340, 1975.

[148] William Snyder, Changhong Mou, Honghu Liu, Omer San, Raffaella De Vita, and Traian Iliescu. Reduced Order Model Closures: A Brief Tutorial. *arXiv:2202.14017 [physics]*, February 2022. arXiv: 2202.14017.

[149] Omer San and Romit Maulik. Neural network closures for nonlinear model order reduction. May 2017.

[150] Gilles Louppe, Joeri Hermans, and Kyle Cranmer. Adversarial Variational Optimization of Non-Differentiable Simulators. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 1438–1447. PMLR, April 2019. ISSN: 2640-3498.

[151] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[152] Charles Audet and Warren Hare. Derivative-free and blackbox optimization. 2017.

[153] Joaquim RRA Martins and Andrew Ning. *Engineering design optimization*. Cambridge University Press, 2021.

[154] Erdem Acar, Gamze Bayrak, Yongsu Jung, Ikjin Lee, Palaniappan Ramu, and Suja Shree Ravichandran. Modeling, analysis, and optimization under uncertainties: a review. *Structural and Multidisciplinary Optimization*, 64(5):2909–2945, 2021.

[155] Maximilian Rixner and Phaedon-Stelios Koutsourelakis. A probabilistic generative model for semi-supervised training of coarse-grained surrogates and enforcing physical constraints through virtual observables. *Journal of Computational Physics*, 434:110218, 2021.

[156] Sebastian Kaltenbach and Phaedon-Stelios Koutsourelakis. Incorporating physical constraints in a deep probabilistic machine learning framework for coarse-graining dynamical systems. *Journal of Computational Physics*, 419:109673, 2020.

[157] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[158] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[159] Ilyes Batatia, Philipp Benner, Yuan Chiang, Alin M Elena, Dávid P Kovács, Janosh Riebesell, Xavier R Advincula, Mark Asta, William J Baldwin, Noam Bernstein, et al. A foundation model for atomistic materials chemistry. *arXiv preprint arXiv:2401.00096*, 2023.

[160] Khaled Saab, Tao Tu, Wei-Hung Weng, Ryutaro Tanno, David Stutz, Ellery Wulczyn, Fan Zhang, Tim Strother, Chunjong Park, Elahe Vedadi, et al. Capabilities of gemini models in medicine. *arXiv preprint arXiv:2404.18416*, 2024.

[161] Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for PDEs. *arXiv preprint arXiv:2405.19101*, 2024.

[162] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[163] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

[164] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

[165] Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural PDE solvers. *arXiv preprint arXiv:2202.03376*, 2022.

[166] Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in Neural Information Processing Systems*, 34:24924–24940, 2021.

[167] Georgios Kissas, Jacob H Seidman, Leonardo Ferreira Guilhoto, Victor M Preciado, George J Pappas, and Paris Perdikaris. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(215):1–63, 2022.

[168] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

[169] Arka Daw, R Quinn Thomas, Cayelan C Carey, Jordan S Read, Alison P Appling, and Anuj Karpatne. Physics-guided architecture (PGA) of neural networks for quantifying uncertainty in lake temperature modeling. In *Proceedings of the 2020 siam international conference on data mining*, pages 532–540. SIAM, 2020.

[170] Nikhil Muralidhar, Jie Bu, Ze Cao, Long He, Naren Ramakrishnan, Danesh Tafti, and Anuj Karpatne. Phynet: Physics guided neural networks for particle drag force prediction in assembly. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 559–567. SIAM, 2020.

[171] William Moses and Valentin Churavy. Instead of rewriting foreign code for machine learning, automatically synthesize fast gradients. *Advances in Neural Information Processing Systems*, 33:12472–12485, 2020.

[172] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[173] Kenji Kawaguchi. On the theory of implicit deep learning: Global convergence with implicit layers. In *International Conference on Learning Representations*, 2020.

[174] R-E Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, 167(2):495–503, 2006.

[175] Lewis F Richardson. *Weather prediction by numerical process.* University Press, 1922.

[176] J.R.R. Tolkien. *The Return of the King*, volume 3 of *The Lord of the Rings*. George Allen & Unwin, London, 1955.

[177] Richard P. Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics*, volume 1-3. Addison-Wesley, Reading, Massachusetts, 1963.

[178] Jiequn Han, Xu-Hui Zhou, and Heng Xiao. VCNN-e: A vector-cloud neural network with equivariance for emulating Reynolds stress transport equations. *arXiv preprint arXiv:2201.01287*, 2022.

[179] Martin Schmelzer, Richard P Dwight, and Paola Cinnella. Discovery of algebraic Reynolds-stress models using sparse symbolic regression. *Flow, Turbulence and Combustion*, 104:579–603, 2020.

[180] Karan Jakhar, Yifei Guan, Rambod Mojgani, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Learning closed-form equations for subgrid-scale closures from high-fidelity data: Promises and challenges. *arXiv preprint arXiv:2306.05014*, 2023.

[181] Wai Tong Chung, Bassem Akoush, Pushan Sharma, Alex Tamkin, Ki Sung Jung, Jacqueline Chen, Jack Guo, Davy Brouzet, Mohsen Talei, Bruno Savard, et al. Turbulence in focus: Benchmarking scaling behavior of 3d volumetric super-resolution with blastnet 2.0 data. *Advances in Neural Information Processing Systems*, 36, 2024.

[182] Yi Li, Eric Perlman, Minping Wan, Yunke Yang, Charles Meneveau, Randal Burns, Shiyi Chen, Alexander Szalay, and Gregory Eyink. A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence*, (9):N31, 2008.

Directory of Supervised Student Research Projects

Within the scope of this dissertation, the following student research projects were developed at the Professorship of Data-Driven Materials Modeling from 2019 to 2024 under the substantial scientific, technical, and content guidance of the author. The author thanks all students for their commitment to supporting this scientific work.

Student	**Research Project**
Leon Riccius	*Machine Learning based approach for investigating Reynolds stress discrepancy based on DNS/LES data*, Master Thesis, 2021. (Incorporated in Paper B [48])
Jonas Eichelsdörfer	*Physics Informed Machine Learning and Hamiltonian Neural Networks*, Master Thesis, 2022.
Mohammad Anas Khan	*Black-box Optimization for engineering systems with score function estimators*, Master Thesis, 2024.

PART III

FULL TEXT OF PUBLICATIONS

A

A probabilistic, data-driven closure model for RANS simulations with aleatoric, model uncertainty

Reprinted from Atul Agrawal and Phaedon-Stelios Koutsourelakis. A probabilistic, data-driven closure model for RANS simulations with aleatoric, model uncertainty. *Journal of Computational Physics*, page 112982, 2024, *licensed under the Creative Commons Attribution License 4.0*

Journal of Computational Physics 508 (2024) 112982

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A probabilistic, data-driven closure model for RANS simulations with aleatoric, model uncertainty

Atul Agrawal [a], Phaedon-Stelios Koutsourelakis [a,b,*]

[a] Technical University of Munich, Professorship of Data-driven Materials Modeling, School of Engineering and Design, Boltzmannstr. 15, 85748 Garching, Germany

[b] Munich Data Science Institute (MDSI - Core member), Garching, Germany

A R T I C L E I N F O

Keywords:
Data-driven turbulence modeling
Reynolds-averaged Navier-Stokes
Uncertainty quantification
Deep neural networks
Differentiable solver

A B S T R A C T

We propose a data-driven, closure model for Reynolds-averaged Navier-Stokes (RANS) simulations that incorporates aleatoric, model uncertainty. The proposed closure consists of two parts. A parametric one, which utilizes previously proposed, neural-network-based tensor basis functions dependent on the rate of strain and rotation tensor invariants. This is complemented by latent, random variables which account for aleatoric model errors. A fully Bayesian formulation is proposed, combined with a sparsity-inducing prior in order to identify regions in the problem domain where the parametric closure is insufficient and where stochastic corrections to the Reynolds stress tensor are needed. Training is performed using sparse, indirect data, such as mean velocities and pressures, in contrast to the majority of alternatives that require direct Reynolds stress data. For inference and learning, a Stochastic Variational Inference scheme is employed, which is based on Monte Carlo estimates of the pertinent objective in conjunction with the reparametrization trick. This necessitates derivatives of the output of the RANS solver, for which we developed an adjoint-based formulation. In this manner, the parametric sensitivities from the differentiable solver can be combined with the built-in, automatic differentiation capability of the neural network library in order to enable an end-to-end differentiable framework. We demonstrate the capability of the proposed model to produce accurate, probabilistic, predictive estimates for all flow quantities, even in regions where model errors are present, on a separated flow in the backward-facing step benchmark problem.

1. Introduction

Turbulence is ubiquitous in fluid flows and of importance to a vast range of applications such as aircraft design, climate and ocean modeling. It has challenged and intrigued scientists and artists for centuries [1]. In the context of the Navier-Stokes equations, the most accurate numerical solution strategy for turbulent flows is offered by Direct Numerical Simulation (DNS), which aims at fully resolving all scales of motion. While this simulation method yields impeccable results, it is prohibitively expensive in terms of computational cost due to the very fine discretizations needed which scale as $\mathcal{O}(\mathrm{Re}^{11/4})$ [2]. Reynolds-averaged Navier-Stokes (RANS) models offer a much more efficient alternative for predicting mean flow quantities. They represent the industry standard

* Corresponding author at: Technical University of Munich, Professorship of Data-driven Materials Modeling, School of Engineering and Design, Boltzmannstr. 15, 85748 Garching, Germany.

E-mail addresses: atul.agrawal@tum.de (A. Agrawal), p.s.koutsourelakis@tum.de (P.-S. Koutsourelakis).

https://doi.org/10.1016/j.jcp.2024.112982
Received 4 July 2023; Received in revised form 23 January 2024; Accepted 29 March 2024

Available online 16 April 2024

which is expected to remain the case in the coming decades [3]. Their predictive accuracy however hinges upon the closure model adopted.

The greater availability of computational resources and the development of scalable learning frameworks in the field of machine learning have had a significant impact in computational fluid mechanics as well [4–6]. Data-driven closures for RANS have revitalized turbulence modeling [7], and a comprehensive reviews can be found in [8,4]. The construction of such closure models consists of two steps: (i) postulating a model form ansatz; and (ii) fitting/learning/inferring model parameters on the basis of the available data. Pertinent approaches have focused on learning model coefficients of a given turbulence model [9] (often with statistical inference), on modeling of correction or source terms for an existing turbulence model [10–14] and on directly modeling the Reynolds stress (RS) tensor [15–18] with symbolic regression [19] or neural networks [15,17,20] or Gaussian Processes [14] or Random Forests [16,18]. Of particular relevance to the present study is the work of [15] wherein they use the non-linear eddy viscosity model (NLEVM) [21] to capture the anisotropic part of the RS tensor using an integrity tensor basis and a deep neural network employing local, invariant flow features. This model owing to its guaranteed Galilean invariance found a wider utilization [17,22,20].

In most of the methods discussed above, data-based training is performed in a *non-intrusive* manner, i.e., without involving the RANS solver in the training process. The major shortcomings of such a strategy (which we attempt to address in the present paper) are two-fold. Firstly inconsistency issues, which can arise between the data-driven model and the baseline turbulence model (e.g., $k-\epsilon$) [23,7]. [24] showed that even substituting RS fields from reputable DNS databases may not lead to satisfactory prediction of the velocity field, and [25] investigated the ill-conditioning that arises in the RANS equations, when employing data-driven models that treat the Reynolds stress as an explicit source term. This ill-conditioning can be amplified within each iteration, thus potentially leading to divergence during the solution procedure. Secondly, such models rely on full-field Reynolds stress training data, which are only available when high-fidelity simulations such as DNS/Large-Eddy Simulations (LES) are used. Unfortunately, such high-fidelity simulations due to their expense are limited to simple geometries and low Reynolds numbers.

In order to address these limitations, we advocate incorporating the RANS model in the training process. This enables one to use indirect data (e.g., mean velocities and pressure) obtained from higher-fidelity simulations or experiments as well as direct data (i.e. RS tensor observables) if this is available. In the subsequent discussions, we will refer to such a training strategy as "model-consistent learning" [7]. It necessitates the solution of a high-dimensional inverse problem that minimizes a discrepancy measure between the RANS solver's output (mean velocities and pressure) and the observables (e.g. mean fields from LES/DNS). As pointed out in [7], model-consistent training or simulation-based Inference [26] benefits from the differentiability of the solver, as it provides derivatives of the outputs with respect to the tunable parameters that can significantly expedite the learning/inference process.

In recent years there has been a concerted effort towards developing differentiable CFD solvers [27–30] in Auto-Differentiation (AD) enabled modules like PyTorch, Tensorflow, JAX, Julia. To the best of the authors' knowledge, this has not been accomplished yet for RANS solvers. One way to enable the computation of parametric sensitivities is by developing adjoint solvers [31,32], which are commonly used in the context of aerodynamic shape optimization [33]. Such adjoint-based modules have also been employed to infer a spatial, corrective field for transport equations [34,10,11] and Reynolds stresses [13]. Recently, [35,36] tried to learn a corrective, multiplicative field in the production term of the Spalart–Allmaras transport model. This is based on an alternative approach outlined in [10], in which empirical correction terms for the turbulence transport equations are learned while retaining a traditional linear eddy viscosity model (LEVM) for the closure. [37] used adjoints to recover a spatially varying eddy viscosity correction factor from sparsely distributed training data, but they also retained the LEVM assumption. More recently, researchers performed model-consistent or CFD-driven training by involving the solver in the training process. [38,39] used gradient-free algorithms to perform symbolic identification of the explicit algebraic Reynolds stress models (ARSM), [40] (with adjoint methods) and [20] (with ensemble methods) combined the RANS solver and a NLEVM-based neural network proposed by [15] in order to learn the model closure. However, they did not account for potential model errors in the closure equation which may arise due to the reasons discussed in the next paragraph.

We argue that even in model-consistent training, a discrepancy in the learnt RS closure model can arise due to the fact that a) the parametric, functional form employed may be insufficient to represent the underlying model,[1] and b) the flow features which are used as input in the closure relation and which are generally restricted to each point in the problem domain (locality/Markovianity assumption [41]), might not contain enough information to predict the optimal RS tensor leading to irrecoverable loss of information. Hence irrespective of the type and amount of training data available, there could be *aleatoric* uncertainty in the closure model that needs to be quantified and propagated in the predictive estimates. We note that much fewer efforts have been directed towards quantifying uncertainties in RANS turbulence models. Earlier, parametric approaches broadly explored the uncertainties in the model choices [9] (i.e., uncertainty involved in choosing the best model among a class of competing models, e.g., $k-\omega, k-\epsilon$) and their respective model coefficients [42]. Recently the shortcomings of the parametric closure models have been recognized by the turbulence modeling community [43,44]. In light of this, various non-parametric approaches have targeted model-form uncertainty whereby uncertainties are directly introduced into the turbulent transport equations or the modeled terms such as the Reynolds stress [22] or eddy viscosity. Such formulations allow for more general estimates of the model inadequacy than the parametric approaches. Researchers have also tried perturbing the eigenvalues [45–47], transport eigenvectors [48] or the tensor invariants. [49] used kernel density estimates to predict the confidence of data-driven models, but it is limited to the prediction of the anisotropic stress and fails to provide any probabilistic bounds. [22] tries to address this issue by incorporating a Bayesian formulation in order to quantify

[1] For example the models based on the Boussinesq hypothesis will fail to capture the flow features driven by the anisotropy of the Reynolds stresses and this intrinsic deficiency cannot be remedied by the calibration of the model coefficients with data.

epistemic uncertainty and then propagating it to quantities of interest like pressure and velocity. For a comprehensive review of modeling uncertainties in the RANS models the reader is directed to [43].

In order to address the aforementioned limitations, we propose a novel probabilistic, model-consistent, data-driven differential framework. The framework enables learning of a NLEVM-based, RS model in a model-consistent way using a differentiable RANS solver, with mean field observables (velocities and/or pressure). To the authors' knowledge, uncertainty quantification has not been addressed for data-driven turbulence model training with indirect observations. We propose to augment the parametric closure model of the RS tensor by a stochastic discrepancy tensor to quantify model errors at different parts of the problem domain. With the introduction of the stochastic discrepancy tensor, we advocate a probabilistic formulation for the associated inverse problem, which provides a superior setting as it is capable of quantifying predictive uncertainties which are unavoidable when any sort of model/dimensionality reduction is pursued and when the model (or its closure) is learned from finite data [50]. To achieve the desired goals, the proposed framework employs the following major elements:

- A discrete, adjoint-based differentiable RANS solver to enable model-consistent, gradient-based learning (Section 2.2, Section 2.2.4).
- The RS closure model consists of a parametric part that is expressed with an invariant neural network as proposed in [15] (Section 2.2.1), to which a *stochastic* discrepancy tensor field is added in order to account for the insufficiency of the parametric part (Section 2.2.2).
- A fully Bayesian formulation that enables the quantification of epistemic uncertainties and their propagation to the predictive estimates (Section 2.2.4, Section 2.2.5)
- This is combined with a *sparsity-inducing* prior model that activates the discrepancy term only in regions of the problem domain where the parametric model is insufficient (Section 2.2.2).

The structure of the rest of the paper is as follows. Section 2 presents the governing equations and their discretization, the closure model proposed consisting of the parametric part and the stochastic corrections provided by latent variables introduced. We also present associated prior and posterior densities, a stochastic Variational Inference scheme that was employed for identifying model parameters and variables as well as the computation of predictive estimates with the trained model. Finally, Section 3 discusses the implementation aspects and demonstrates the accuracy and efficacy of the proposed framework in the backward-facing step test case [51], where the linear eddy viscosity models are known to fail. We compare our results with LES reference values and the $k-\epsilon$ model, which is arguably the most commonly used RANS model. In Section 4, we summarize our findings and discuss limitations and potential enhancements.

2. Methodology

2.1. Problem statement

2.1.1. Reynolds-Averaged Navier-Stokes (RANS) equations

The Navier-Stokes equations for incompressible flows of Newtonian fluids are given by (in indicial notation):

$$\frac{\partial U_i}{\partial t} + \frac{\partial}{\partial x_j}(U_i U_j) = \nu \frac{\partial^2 U_i}{\partial x_j \partial x_j} - \frac{1}{\rho}\frac{\partial P}{\partial x_i}, \tag{1}$$

$$\frac{\partial U_j}{\partial x_j} = 0, \tag{2}$$

where i, j are free and dummy indices respectively taking values $1, 2, 3$ and U_i, P, t, x_j, ν and ρ represent the flow velocity, pressure, time, spatial coordinates, the dynamic viscosity and the density of the fluid respectively. The non-linearity of the convective term $\frac{\partial}{\partial x_j}(U_i U_j)$ gives rise to chaotic solutions beyond a critical value of the *Reynolds number* Re. This necessitates very fine spatio-temporal discretizations in order to capture the salient scales. Such brute-force, fully-resolved simulations, commonly referred to as Direct Numerical Simulations (DNS), can become prohibitively expensive, particularly as Re increases.

The velocity field can be decomposed into its time-averaged (or mean) part u and the part corresponding, to generally fast, fluctuations $\tilde{u}$ as:

$$U_i(\boldsymbol{x}, t) = u_i(\boldsymbol{x}) + \tilde{u}_i(\boldsymbol{x}, t), \tag{3}$$

$$\text{where,} \quad u_i(\boldsymbol{x}) = \langle U_i(\boldsymbol{x}, t)\rangle = \lim_{T\to\infty} \frac{1}{T}\int_0^T U_i(\boldsymbol{x}, t)\, dt. \tag{4}$$

Similarly the pressure field is also decomposed as

$$P(\boldsymbol{x}, t) = p(\boldsymbol{x}) + \tilde{p}(\boldsymbol{x}, t), \tag{5}$$

$$\text{where} \quad p(\boldsymbol{x}) = \langle P(\boldsymbol{x},t) \rangle = \lim_{T\to\infty} \frac{1}{T} \int_0^T P(\boldsymbol{x},t)dt. \tag{6}$$

Substituting these decompositions into the Navier-Stokes equations (Equation (1)) and applying time-averaging results in the Reynolds-averaged Navier-Stokes (RANS) equations [2,52], i.e.:

$$u_j \frac{\partial u_i}{\partial x_j} - \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} + \frac{1}{\rho}\frac{\partial p}{\partial x_i} = -\frac{\partial \left\langle \tilde{u}_i \tilde{u}_j \right\rangle}{\partial x_j}, \tag{7}$$

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{8}$$

where $\langle \cdot \rangle$ denotes the time average of the arguments as in Equation (4) or Equation (6). In several engineering applications involving turbulent flows, the quantities of interest depend upon the time-averaged quantities. These can be obtained by solving the RANS equations which in general implies a much lower computational cost than DNS.

2.1.2. *The closure problem*

The RANS equations are unfortunately *unclosed* as they depend on the cross-correlation of the fluctuating velocity components, commonly referred to as the Reynolds-Stress (RS) tensor $\boldsymbol{\tau}_{RS}$:

$$\boldsymbol{\tau}_{RS} = -\left\langle \tilde{u}_i \tilde{u}_j \right\rangle. \tag{9}$$

The goal of pertinent efforts is therefore to devise appropriate closure models where the RS tensor $\boldsymbol{\tau}_{RS}$ is expressed as a function as the primary state variables in the RANS equations i.e. the time-averaged flow quantities. Closure models are of three types: (i) Functional, which use physical insight to construct the closure; (ii) Structural, which use mathematical tools; and (iii) Data-driven, which employ experimental/simulation data [53]. For a comprehensive review, the reader is directed to [54,53,55]. Classically, turbulence models are devised to represent higher-order moments of the velocity fluctuations in terms of lower-order moments. This can be done directly, as in the case of the eddy-viscosity models, or indirectly, as in the case of models based on the solution of additional partial differential equations [2].

The most commonly employed strategy is based on the linear-eddy-viscosity-model (LEVM), which uses the Boussinesq approximation according to which $\boldsymbol{\tau}_{RS}$ is expressed as:

$$\boldsymbol{\tau}_{LEVM} = \frac{2}{3}k\boldsymbol{I} - 2\nu_t \bar{\boldsymbol{S}}, \tag{10}$$

where ν_t is the eddy viscosity, $\bar{\boldsymbol{S}} = \frac{1}{2}\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right)$ is the mean strain-rate tensor, $\boldsymbol{I}$ is the second order identity tensor, and $k = -\frac{1}{2}\mathrm{tr}(\boldsymbol{\tau}_{RS})$ is the turbulent kinetic energy. The eddy viscosity is computed after solving the equation(s) for the turbulent flow quantities such as the turbulent kinetic energy k and the turbulent energy dissipation ϵ (e.g., the $k-\epsilon$ model [56]), or the specific dissipation ω (e.g., the $k-\omega$ [57]). Although the Boussinesq approximation provides accurate results for a range of flows, it can give rise to predictive inaccuracies which are particularly prominent when trying to capture flows with significant curvatures, recirculation zones, separation, reattachment, anisotropy, etc [2,58]. Attempts to overcome this weakness have been made in the form of nonlinear eddy viscosity models (e.g., [59,21,60]), Reynolds-stress transport models (e.g., [61]) and ARSM (e.g., [62,63]). These models have not received widespread attention because they lack the robustness of LEVM and involve more parameters that need to be calibrated.

2.2. *Probabilistic, model-consistent data-driven differential framework*

Upon discretization using e.g., a finite element scheme (Appendix A), one can express the RANS equations (Equation (7)) in residual form as:

$$\mathcal{G}(\boldsymbol{z}) = \boldsymbol{B}\boldsymbol{\tau} \tag{11}$$

$$\text{or,} \quad \mathcal{R}(\boldsymbol{z};\boldsymbol{\tau}) := \mathcal{G}(\boldsymbol{z}) - \boldsymbol{B}\boldsymbol{\tau} = 0, \tag{12}$$

where $\boldsymbol{z} = [\boldsymbol{u}, p]^T$ summarily denotes the discretized velocity $\boldsymbol{u}$ and pressure p fields and $\boldsymbol{\tau}$ the discretized RS field. E.g. for a two-dimensional flow domain $\boldsymbol{z} \in \mathbb{R}^{N\times 3}$, $\boldsymbol{\tau} \in \mathbb{R}^{N\times 3}$ where N is the number of grid points. The discretization scheme employed and other implementation details are discussed in Appendix A. We denote with $\mathcal{G}$ the discretized, non-linear operator accounting for the advective and diffusive terms on the left-hand side of Equation (7) as well as the conservation of mass in Equation (8), and with $\boldsymbol{B}$ the matrix (i.e. linear operator) arising from the divergence term on the right-hand side of Equation (7).

Traditional, data-driven strategies postulate a closure e.g. $\boldsymbol{\tau}_{\theta}(\boldsymbol{z})$ (or most often $\boldsymbol{\tau}_{\theta}(\boldsymbol{u})$) dependent on some tunable parameters $\boldsymbol{\theta}$, which they determine either by assuming that reference Reynolds-stress data is available from DNS simulations (or in general, from higher-fidelity models such as LES) or by employing experimental or simulation-based data of the mean velocities/pressures i.e. of $\boldsymbol{z}$. The former scenario which is referred to as *model regression* [64] has received significant attention in the past (e.q. [22,15,65,17]). Apart from the heavier data requirements, it does not guarantee that the trained model would yield accurate predictions of $\boldsymbol{z}$ [24] as even small errors in $\boldsymbol{\tau}$ might get amplified when solving Equation (12). The second setting, referred to as *trajectory regression* in

[64], might be able to make use of indirect and noisy observations but is much more cumbersome as repeated model evaluations and parametric sensitivities, i.e. a differentiable solver, are needed for training.

Critical to any data-driven model is its ability to generalize i.e. to produce accurate predictions under different flow scenarios. On one hand this depends on the training data available but on the other, on incorporating a priori available domain knowledge. The latter can attain various forms and certainly includes known invariances or equivariances that characterize the associated maps. Apart from this and the particulars of the parameterized model form, a critical aspect pertains to uncertainty quantification. We distinguish between parametric and model uncertainty. The former is of epistemic origin and has been extensively studied (e.g., [9,42,43]). Bayesian formulations offer a rigorous manner for quantifying it and ultimately propagating it in the predictive estimates in the form of the predictive posterior. We note however that in the limit of infinite data, the posterior of the model parameters $\boldsymbol{\theta}$ (no matter what these are or represent) would collapse to a Dirac-delta i.e. point-estimates for $\boldsymbol{\theta}$ would be obtained. This false lack of uncertainty does not imply that the model employed is perfect as the true (unknown) closure might attain a form not contained in the parametric family used or in the features of $\boldsymbol{z}$ that appear in the input (e.g. even though all models proposed employ a locality assumption in the closure equations, non-local features of $\boldsymbol{u}$ might be needed).

The issue of model uncertainty in the closure equations which is of an aleatoric nature, has been much less studied and represents the main contribution of this work. In particular, we augment the parametric closure model $\boldsymbol{\tau}_{\theta}(\boldsymbol{u})$ with a set of latent (i.e. unobserved) random variables $\boldsymbol{\epsilon}_{\tau}$ which are embedded in the model equations and which quantify model discrepancies at each grid point. In reference to the discretized RS vector $\boldsymbol{\tau}$ in Equation (12), we propose:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\theta}(\boldsymbol{u}) + \boldsymbol{\epsilon}_{\tau}. \tag{13}$$

We emphasize the difference between model parameters $\boldsymbol{\theta}$ and the random variables $\boldsymbol{\epsilon}_{\tau}$. While both are informed by the data, the latter remain random even in the limit of infinite data. As we explain in the sequel, we advocate a fully Bayesian formulation that employs indirect observations of the velocities/pressures. These are combined with appropriate sparsity-inducing priors which can turn-off model discrepancy terms when the parametric model is deemed to provide an adequate fit. In this manner, the regions of the problem domain where the closure is most problematic are identified while probabilistic, predictive estimates are always obtained. In particular, in Section 2.2.1 the parametric part of the closure model is discussed. In Section 2.2.2 the proposed, stochastic, discrepancy tensor is presented. In Section 2.2.3 prior and posterior densities are discussed and in Section 2.2.4 the corresponding inference and learning algorithms are introduced. Finally in Section 2.2.5, the computation of predictive estimates using the trained model is discussed.

2.2.1. Parametric RS model

In this section we discuss the parametric part, i.e. $\boldsymbol{\tau}_{\theta}(\boldsymbol{u})$ in the closure model of Equation (13). As this represents a vector containing its values at various grid points over the problem domain, the ensuing discussion and equations should be interpreted as per grid point. We note that the most popular LEVM model (Equation (10)) assumes that the anisotropic part of the $\boldsymbol{\tau}_{LEVM}$, is linearly related to the mean strain rate tensor $\bar{\boldsymbol{S}}$. This linear relation assumption restricts the model to attain a small subset of all the possible states of turbulence. This subset is referred to as the plane strain line [66]. Experimental and DNS data show that turbulent flows explore large regions of the domain of realizable turbulence states.

In the present work, we make use of the invariant neural network architecture proposed by [15] which relates the anisotropic part of the RS tensor with the symmetric and antisymmetric components of the velocity gradient tensor. By using tensor invariants, the neural network is able to achieve both Galilean invariance as well as rotational invariance. The Navier-Stokes equations are Galilean-invariant, i.e. they remain unchanged for all inertial frames of reference. The theoretical foundation of this neural network lies in the Non-Linear Eddy Viscosity Model (NLEVM) proposed by [21] and has been used in several studies [22,13,17]. By employing barycentric realizability maps [46,45,67,66,48], it was shown in [17] that this architecture overcomes the plane strain line restriction and can explore other realizable states.

In the model proposed by [21], the normalized anisotropic tensor of the R-S was given by $\boldsymbol{b} := \boldsymbol{b}(\boldsymbol{S}, \boldsymbol{\Omega})$, which was a function of the normalized mean rate of strain tensor $\boldsymbol{S}$ and the rotation tensor $\boldsymbol{\Omega}$, i.e.:

$$\boldsymbol{\tau} = 2k\boldsymbol{b} + \frac{2k}{3}\boldsymbol{I}; \quad \boldsymbol{S} = \frac{1}{2}\frac{k}{\epsilon}\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right) \quad \boldsymbol{\Omega} = \frac{1}{2}\frac{k}{\epsilon}\left(\nabla\boldsymbol{u} - \nabla\boldsymbol{u}^T\right). \tag{14}$$

Through the application of Cayley-Hamilton theorem [21], the following general expression for the anisotropy tensor $\boldsymbol{b}$ was adopted:

$$\boldsymbol{b} = \sum_{k=1}^{10} G^{(k)}(\underbrace{\mathcal{I}_1 ... \mathcal{I}_5}_{\substack{Scalar\\Invariants}})\boldsymbol{\mathcal{T}}^{(k)}, \tag{15}$$

where:

$$\mathcal{I}_1 = \mathrm{tr}(\boldsymbol{S}^2), \quad \mathcal{I}_2 = \mathrm{tr}(\boldsymbol{\Omega}^2), \quad \mathcal{I}_3 = \mathrm{tr}(\boldsymbol{S}^3), \quad \mathcal{I}_4 = \mathrm{tr}(\boldsymbol{\Omega}^2\boldsymbol{S}), \quad \mathcal{I}_5 = \mathrm{tr}(\boldsymbol{\Omega}^2\boldsymbol{S}^2), \tag{16}$$

and $\boldsymbol{\mathcal{T}}^{(k)}$ are the symmetric tensor basis functions (the complete set is listed in Table 1). The coefficients $G^{(i)}$ are scalar, non-linear functions which depend on the five invariants $\mathcal{I}_1 ... \mathcal{I}_5$ and must be determined. If $G^{(1)} = -0.09, G^{(n)} = 0$, the NLEVM degenerates to the classical $k - \epsilon$ model. When the NLEVM was proposed, it was impossible to find good approximations for these functions and as

Table 1
Complete set of basis tensors $\mathcal{T}^{(n)}$, that can be formed form $\boldsymbol{S}$ and $\boldsymbol{\Omega}$. Matrix notation is used for clarity. The trace of a tensor is denoted by $\mathrm{tr}(\boldsymbol{S}) = S_{ii}$.

$\mathcal{T}^{(1)}$	$= \boldsymbol{S}$,	$\mathcal{T}^{(6)}$	$= \boldsymbol{\Omega}^2\boldsymbol{S} + \boldsymbol{S}\boldsymbol{\Omega}^2 - \frac{2}{3}\mathrm{tr}(\boldsymbol{S}\boldsymbol{\Omega}^2)\boldsymbol{I}$,
$\mathcal{T}^{(2)}$	$= \boldsymbol{S}\boldsymbol{\Omega} - \boldsymbol{\Omega}\boldsymbol{S}$,	$\mathcal{T}^{(7)}$	$= \boldsymbol{\Omega}\boldsymbol{S}\boldsymbol{\Omega}^2 + \boldsymbol{\Omega}^2\boldsymbol{S}\boldsymbol{\Omega}$,
$\mathcal{T}^{(3)}$	$= \boldsymbol{S}^2 - \frac{1}{3}\mathrm{tr}(\boldsymbol{S}^2)\boldsymbol{I}$,	$\mathcal{T}^{(8)}$	$= \boldsymbol{S}\boldsymbol{\Omega}\boldsymbol{S}^2 - \boldsymbol{S}^2\boldsymbol{\Omega}\boldsymbol{S}$,
$\mathcal{T}^{(4)}$	$= \boldsymbol{\Omega}^2 - \frac{1}{3}\mathrm{tr}(\boldsymbol{\Omega}^2)\boldsymbol{I}$,	$\mathcal{T}^{(9)}$	$= \boldsymbol{\Omega}^2\boldsymbol{S}^2 + \boldsymbol{S}^2\boldsymbol{\Omega}^2 - \frac{2}{3}\mathrm{tr}(\boldsymbol{S}^2\boldsymbol{\Omega}^2)\boldsymbol{I}$,
$\mathcal{T}^{(5)}$	$= \boldsymbol{\Omega}\boldsymbol{S}^2 - \boldsymbol{S}^2\boldsymbol{\Omega}$,	$\mathcal{T}^{(10)}$	$= \boldsymbol{\Omega}\boldsymbol{S}^2\boldsymbol{\Omega}^2 - \boldsymbol{\Omega}^2\boldsymbol{S}^2\boldsymbol{\Omega}$.

a result, it did not receive adequate attention. This hurdle however was overcome with the help of machine learning [15] where $G^{(i)}$ were learned from high-fidelity simulation data. Neural networks with parameters $\boldsymbol{\theta}$ were employed for the coefficients i.e. $G_{\boldsymbol{\theta}}^{(i)}$ and:

$$\boldsymbol{b}_{\boldsymbol{\theta}} = \sum_{i=1}^{10} G_{\boldsymbol{\theta}}^{(i)}(\underbrace{\mathcal{I}_1 \dots \mathcal{I}_5}_{\substack{Scalar\\ Invariants}})\mathcal{T}^{(i)}; \quad \boldsymbol{\tau}_{\boldsymbol{\theta}} = 2k\boldsymbol{b}_{\boldsymbol{\theta}} + \frac{2k}{3}\boldsymbol{I}. \tag{17}$$

We use $\boldsymbol{\tau}_{\boldsymbol{\theta}}$ to denote the neural-network-based, discretized RS tensor terms in the subsequent discussions.

As in [22], we employ the following prior for the NN parameters $\boldsymbol{\theta}$:

$$p(\boldsymbol{\theta} \mid \nu) = \mathcal{N}(\boldsymbol{\theta}|0, \nu^{-1}\boldsymbol{I}_{d_{\theta}}), \quad p(\nu) = Gamma(\nu|a_0, b_0), \tag{18}$$

where $d_{\theta} = dim(\boldsymbol{\theta})$ and a Gamma hyperprior was used for the common precision hyperparameter ν with $(a_0, b_0) = (1.0, 0.02)$. The resulting prior has the density of a Student's $\mathcal{T}$-distribution centered at zero, which is obtained by analytically marginalizing over the hyperparameter ν [68].

Similarly however to the most widely used RANS closure models, such as the Launder-Sharma $k-\epsilon$ [56] or Wilcox's $k-\omega$ [57], which are based on the Boussinesq turbulent-viscosity hypothesis, the present model of $\boldsymbol{\tau}_{\boldsymbol{\theta}}$ postulates that the RS tensor at each point in the problem domain depends on the flow features at the same point (locality assumption). This is a very strong assumption for flows that exhibit strong inhomogeneity [2].

2.2.2. Stochastic discrepancy term to RS

We argue that despite the careful selection of input features of the mean velocity field and the flexibility in the resulting map afforded by the NN architecture, the final form might not be able to accurately capture the true RS or at least not at every grid point in the problem domain. This would be the case, if e.g. non-local terms, which are unaccounted in the aforementioned formulation, played a significant role. As mentioned earlier, this gives rise to model uncertainty of an aleatoric nature which is of a different type than the epistemic uncertainty in the parameters $\boldsymbol{\theta}$ of the closure model presented in the previous section. It is this model uncertainty that we propose to capture with the latent, random vector $\boldsymbol{\epsilon}_{\tau}$ in Equation (13). As for $\boldsymbol{\tau}_{\boldsymbol{\theta}}$ in the previous section, $\boldsymbol{\epsilon}_{\tau}$ is a vector containing the contribution from all grid points in the problem domain. Hence, in the two-dimensional setting and given the symmetry of the RS tensor, $\boldsymbol{\epsilon}_{\tau}$ would be of dimension $d_{\epsilon} = 3N$ where N is the total number of grid points.

Before discussing the prior specification for $\boldsymbol{\epsilon}_{\tau}$ and associated inference procedures, we propose a dimension-reduced representation that would facilitate subsequent tasks given the high values that N takes in most simulations. In particular, we represent $\boldsymbol{\epsilon}_{\tau}$ as:

$$\boldsymbol{\epsilon}_{\tau} = \boldsymbol{W}\boldsymbol{E}_{\tau}. \tag{19}$$

It is based on considering N_d subdomains of the problem domain and assuming that for all grid points in a certain subdomain, the corresponding RS discrepancy terms are identical. This can be expressed as in Equation (19) above where the entries of $\boldsymbol{W}$ are 1 if a corresponding grid point (row of $\boldsymbol{W}$) belongs in a certain subdomain (column of $\boldsymbol{W}$) and 0 otherwise. A non-binary $\boldsymbol{W}$ would also be possible, although in this case its physical interpretation in terms of subdomains would be occluded. The vector $\boldsymbol{E}_{\tau} = \{\boldsymbol{E}_{\tau,J}\}_{J=1}^{N_d}$ contains therefore the RS discrepancy terms for each subdomain, e.g. $dim(\boldsymbol{E}_{\tau,J}) = 3$ for a two-dimensional flow. The dimensionality reduction scheme resembles Principal Component Analysis (PCA) [69]. In contrast to the latter however, we never have data/observations of the vectors to be reduced, i.e. $\boldsymbol{\epsilon}_{\tau}$ in our case. These are inferred implicitly from the LES data and simultaneously with $\boldsymbol{W}$. In the ensuing numerical illustrations, the division into subdomains is done in a regular manner and $\boldsymbol{W}$ is prescribed a priori. One could nevertheless readily envision a learnable $\boldsymbol{W}$ or even an adaptive refinement into subdomains.

The incorporation of the model discrepancy variables $\boldsymbol{\epsilon}_{\tau}$ or $\boldsymbol{E}_{\tau}$ at each grid-point or subdomain introduces redundancies i.e. there would be an infinity of combinations of $\boldsymbol{\theta}$ and $\boldsymbol{\epsilon}_{\tau}/\boldsymbol{E}_{\tau}$ that could fit the data equally well. In order to address this issue, we invoke the concept of *sparsity* which has been employed in similar situations in the context of physical modeling [70,71]. To this end, we make use of a sparsity-enforcing Bayesian prior based on the Automatic Relevance Determination (ARD) [72,73]. In particular:

$$p(\boldsymbol{E}_{\tau}|\boldsymbol{\Lambda}) = \prod_{J=1}^{N_d} p(\boldsymbol{E}_{\tau,J}|\boldsymbol{\Lambda}^{(J)}) = \prod_{J=1}^{N_d} \mathcal{N}\left(\boldsymbol{E}_{\tau,J} \mid \boldsymbol{0},\ diag(\boldsymbol{\Lambda}_J)^{-1}\right), \tag{20}$$

where $\boldsymbol{E}_{\tau,J}$ denotes the stochastic RS discrepancy term at subdomain J and the vector of hyperparameters $\boldsymbol{\Lambda}_J$ contains the corresponding precisions (e.g. for two-dimensional flows, $dim(\boldsymbol{\Lambda}_J) = 3$). A-priori therefore we assume that the RS discrepancies are zero on average with an unknown variance/precision that will be learned from the data as it will be discussed in the sequel. This is combined with the following hyperprior (omitting the hyperparameters α_0, β_0):

$$p(\boldsymbol{\Lambda}) = \prod_{J=1}^{N_d} \prod_{\ell=1}^{L} Gamma(\Lambda_{J,\ell} \,|\alpha_0, \beta_0), \tag{21}$$

where $\Lambda_{J,\ell}$ denotes the ℓ^{th} entry (e.g. $L = 3$ for two-dimensional flows) of the vector of precision hyperparameters in subdomain J. We note that when $\Lambda_{J,\ell} \to \infty$, then the corresponding model discrepancy term $E_{\tau,J,\ell} \to 0$. The resulting prior for $\boldsymbol{E}_\tau$ arising by marginalizing the hyperparameters $\boldsymbol{\Lambda}$ is a light-tailed, Student's t-distribution [74] that promotes solutions in the vicinity of 0 unless strong evidence in the data suggests otherwise. The hyperparameters α_0, β_0 are effectively the only ones that need to be provided by the analyst. We advocate very small values ($\alpha_0 = \beta_0 = 10^{-3}$ was used in the ensuing numerical illustrations) which correspond to an uninformative prior [75]. This is consistent with the values adopted in the original paper where the ARD prior was employed in conjunction with variational inference i.e. in [75].

The dimension of the $\boldsymbol{\epsilon}_\tau$ is high i.e. $3N$ in 2D (and $6N$ in 3D) where N is the total number of grid points. In contrast, the dimension of $\boldsymbol{\theta}$ is independent of the grid as the parametric part of the closure model is the same at all grid points. Depending on the amount of data, one can envision cases where multiple combinations of $\boldsymbol{\epsilon}_\tau$ and $\boldsymbol{\theta}$ could fit the data equally well. Non-zero values of $\boldsymbol{\epsilon}_\tau$ would imply model errors at the corresponding grid points. Ceteris paribus, we would prefer the combination with the least number of model errors or equivalently the one for which our parametric model can capture for most of the domain the closure accurately. It is indeed these types of solutions/combinations that the ARD prior promotes.

2.2.3. *Data, likelihood and posterior*

The probabilistic model proposed is trained with *indirect* observational data that pertain to time-averaged velocities and pressures at various points in the problem domain. This is in contrast to the majority of efforts in data-driven RANS closure modeling [22,15,65,17], which employ *direct* RS data. In the ensuing numerical illustrations, the data is obtained from higher-fidelity computational simulations, but one could readily make use of actual, experimental observations.

In particular, we consider $M \geq 1$ flow settings and denote the observations collected as $\mathcal{D} = \{\hat{\boldsymbol{z}}^{(m)}\}_{m=1}^{M}$. These consist of time-averaged velocity/pressure values where $dim(\hat{\boldsymbol{z}}^{(m)}) = N_{obs}$. The locations of these measurements do not necessarily coincide with the mesh used to solve the RANS model in Equation (7) nor is it necessary that the same number of observations is available for each of the M flow settings. The data is ingested with the help of a Gaussian likelihood:

$$\begin{aligned} p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{\epsilon}_\tau^{(1:M)}) &= \textstyle\prod_{m=1}^{M} p(\hat{\boldsymbol{z}}^{(m)} \mid \boldsymbol{\theta}, \boldsymbol{\epsilon}_\tau^{(m)}) \\ &= \textstyle\prod_{m=1}^{M} \mathcal{N}(\hat{\boldsymbol{z}}^{(m)} \mid \boldsymbol{z}(\boldsymbol{\theta}, \boldsymbol{\epsilon}_\tau^{(m)}), \boldsymbol{\Sigma}), \end{aligned} \tag{22}$$

where $\boldsymbol{z}(\boldsymbol{\theta}, \boldsymbol{\epsilon}_\tau^{(m)})$ denotes the solution vector of the discretized RANS equations (see Equation (7)) with the closure model suggested by Equation (13) i.e. $\boldsymbol{\tau} = \boldsymbol{\tau}_{\boldsymbol{\theta}}(\boldsymbol{u}) + \boldsymbol{\epsilon}_\tau^{(m)}$. We note that a different set of latent variables $\boldsymbol{\epsilon}_\tau^{(m)}$ is needed for each flow scenario as, by its nature, model discrepancy will in general assume different values under different flow conditions. We also note that the $\boldsymbol{\epsilon}_\tau$ is grid-dependent or problem geometry-dependent, which restricts it from making predictions for a completely different flow geometry. One could, with appropriate modeling enhancements, learn the dependence of $\boldsymbol{\epsilon}_\tau$ on flow parameters (e.g. boundary/initial conditions, geometry, Re number) which would make it usable in other flow geometries. Alternatively, it could be trained under different flow settings (e.g. different Re numbers as we do) and learn, on aggregate, the model error which in turn could be used to make predictions under different flow settings (as in our case for different Re number).

We denote with $\boldsymbol{\Sigma}$ the covariance matrix of the Gaussian likelihood which, given the absence of actual observation noise, plays the role of a tolerance parameter determining the tightness of the fit. The covariance was expressed as $\boldsymbol{\Sigma} = diag(\sigma_1^2, \cdots, \sigma_{3N_{obs}}^2)$, where the values in the diagonal vector are set to 1% of the mean of the squares of each observable across the M flow settings i.e. $\sigma_i^2 = 0.01\, \frac{1}{M} \sum_{m=1}^{M} \left(\hat{z}_i^{(m)}\right)^2$.

By combining the likelihood above with the priors discussed in the previous sections as well as by employing the dimensionality reduction scheme of Equation (19) according to which $\boldsymbol{\epsilon}_\tau^{(m)}$ can be expressed as $\boldsymbol{\epsilon}_\tau^{(m)} = \boldsymbol{W}\boldsymbol{E}_\tau^{(m)}$, we obtain a posterior on:

- the parameters $\boldsymbol{\theta}$ of the parametric closure model,
- the latent variables $\boldsymbol{E}_\tau^{(1:M)}$ expressing the stochastic model discrepancy in *each* of the M training conditions,
- the hyperparameters $\boldsymbol{\Lambda}$ in the hyperprior of $\boldsymbol{E}_\tau^{(1:M)}$,

which would be of the form (omitting given hyperparameters):

$$\begin{aligned} p(\boldsymbol{\theta}, \boldsymbol{E}_\tau^{(1:M)}, \boldsymbol{\Lambda} \mid \mathcal{D}) &\propto p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{E}_\tau^{(1:M)})\; p(\boldsymbol{E}_\tau^{(1:M)}|\boldsymbol{\Lambda})\; p(\boldsymbol{\theta})\; p(\boldsymbol{\Lambda}) \\ &= \left(\textstyle\prod_{m=1}^{M} p(\hat{\boldsymbol{z}}^{(m)} \mid \boldsymbol{\theta}, \boldsymbol{E}_\tau^{(m)})\; p(\boldsymbol{E}_\tau^{(m)}|\boldsymbol{\Lambda}) \right)\; p(\boldsymbol{\theta})\; p(\boldsymbol{\Lambda}). \end{aligned} \tag{23}$$

An illustration of the corresponding graphical model is contained in Fig. 1.

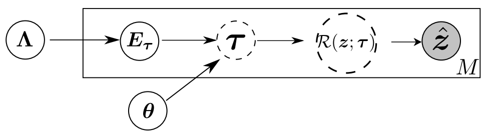


Fig. 1. Probabilistic graphical model of the proposed model including model parameters (θ, Λ), latent variables (E_τ) and observables $\hat{z}$ from M flow scenarios. Deterministic nodes are indicated with circles with dashed line, stochastic with circles with solid line and known/observed are shaded.

2.2.4. Inference and learning

On the basis of the probabilistic model proposed and the posterior formulated in the previous section, we discuss numerical inference strategies for identifying the unknown parameters and latent variables. The intractability of the posterior stems from the likelihood which entails the solution of the discretized RANS equations. We advocate the use of Stochastic Variational Inference (SVI) [76] which results in a closed-form approximation of the posterior $p(\theta, E_\tau^{(1:M)}, \Lambda|\mathcal{D})$. In contrast to the popular, sampling-based strategies (MCMC, SMC etc.), SVI yields biased estimates at the benefit of computational efficiency. Readers interested in comparative studies/discussions on SVI and MCMC in terms of accuracy and computational efficiency are directed to [77]. Given a family of probability densities $q_\xi\left(\theta, \Lambda, E_\tau^{(1:M)}\right)$ parametrized by ξ, we find the optimal, i.e. the one that is closest to the exact posterior in terms of their Kullback-Leibler divergence, by maximizing the Evidence Lower Bound (ELBO) $\mathcal{F}(\xi)$ [68]:

$$\begin{aligned}\mathcal{F}(\xi) &= \mathbb{E}_{q_\xi\left(\theta,\Lambda,E_\tau^{(1:M)}\right)}\left[\log\left(\frac{p\left(\mathcal{D},\theta,\Lambda,E_\tau^{(1:M)}\right)}{q_\xi\left(\theta,\Lambda,E_\tau^{(1:M)}\right)}\right)\right] \\ &= \mathbb{E}_{q_\xi\left(\theta,\Lambda,E_\tau^{(1:M)}\right)}\left[\log\left(\frac{p(\mathcal{D} \mid \theta, E_\tau^{(1:M)})\, p(E_\tau^{(1:M)}|\Lambda)\, p(\theta)\, p(\Lambda)}{q_\xi\left(\theta,\Lambda,E_\tau^{(1:M)}\right)}\right)\right].\end{aligned} \tag{24}$$

As its name suggests, it can be readily shown and that ELBO lower-bounds the model log-evidence and their difference is given by the aforementioned KL-divergence i.e.:

$$\log p(\mathcal{D}) = \mathcal{F}(\xi) + KL\left(q_\xi(\theta,\Lambda,E_\tau^{(1:M)}) \,\middle\|\, p(\theta,\Lambda,E_\tau^{(1:M)}|\mathcal{D})\right). \tag{25}$$

In order to expedite computations we employ a mean-field assumption [77] according to which the approximate posterior is factorized as:

$$q_\xi\left(\theta,\Lambda,E_\tau^{(1:M)}\right) = q_\xi(\theta) q_\xi(\Lambda) \prod_{m=1}^{M} q_\xi\left(E_\tau^{(m)}\right). \tag{26}$$

We employ Dirac-deltas for the first two densities i.e.:

$$q_\xi(\theta) = \delta(\theta - \theta_{MAP}), \tag{27}$$

$$q_\xi(\Lambda) = \delta(\Lambda - \Lambda_{MAP}). \tag{28}$$

In essence, we obtain point estimates for θ, Λ which coincide with the Maximum-A-Posteriori (MAP) estimates. The specific choice of Dirac-deltas was motivated by computational cost reasons in the inference and prediction steps. Furthermore, given that the dimension of θ is generally (much) smaller than the number of observations (see section 3), we anticipate that its posterior would not deviate much from a Dirac-delta. The final reason for which posterior uncertainty (albeit small) on θ was not included, was to emphasize the significance and impact of the stochastic, model correction term that we propose.

For the model discrepancy variables $E_\tau^{(m)}$, we employ uncorrelated Gaussians given by:

$$q_\xi(E_\tau^{(m)}) = \mathcal{N}(E_\tau^{(m)} \mid \mu_E^{(m)}, \mathrm{diag}(\sigma_E^{2,(m)})), \quad \forall i \in \{1,\dots,M\}. \tag{29}$$

In summary, the vector ξ of the parameters in the variational approximation consists of:

$$\xi = \{\theta_{MAP}, \Lambda_{MAP}, \{\mu_E^{(m)}, \sigma_E^{2,(m)}\}_{m=1}^{M}\}. \tag{30}$$

The updates of the parameters ξ are carried out using derivatives of the ELBO. These entail expectations with respect to q_ξ which are estimated (with noise) by Monte Carlo in conjunction with the ADAM stochastic optimization scheme [78]. In order to reduce the Monte-Carlo noise in the estimates, we employ the reparametrization trick [79]. This is made possible here due to the form of the approximate posterior q_ξ. In particular, if we summarily denote with $\eta = \{\theta, \Lambda, E_\tau^{(1:M)}\}$ and given that the approximate posterior $q_\xi(\eta)$ can be represented by deterministic transform $\eta = g_\xi(\phi)$, where ϕ follows a known density $q(\phi)$,[2] the expectations involved in the ELBO and, more importantly, in its gradient can be rewritten as:

$$\nabla_\xi \mathcal{F}(\xi) = \mathbb{E}_{q(\phi)}\left[\nabla_\xi g_\xi(\phi) \nabla_\eta \left(\log p(\mathcal{D},\eta) - \log q_\xi(\eta)\right)\right]. \tag{31}$$

[2] Based on the form of q_ξ in Equations (26), (27), (28), (29), the transform employed can be written as $\theta = \theta_{MAP}$, $\Lambda = \Lambda_{MAP}$ and $E_\tau^{(m)} = \mu_E^{(m)} + \mathrm{diag}(\sigma_E^{2,(m)})\, \phi$ where $q(\phi) = \mathcal{N}(\phi|\, \mathbf{0}, \boldsymbol{I})$.

Algorithm 1: Inference and learning using SVI.

```
   Input  : $\mathcal{D} = \{\hat{z}^{(i)}\}_{m=1}^{M}$, $\boldsymbol{W}$
   Output: $\xi = \{\theta_{MAP}, \boldsymbol{\Lambda}_{MAP}, \{\boldsymbol{\mu}_E^{(m)}, \boldsymbol{\sigma}_E^{2,(m)}\}_{m=1}^{M}\}$
1  while ELBO $\hat{\mathcal{F}}$ not converged do
2      $\theta \leftarrow \theta_{MAP}$,   $\boldsymbol{\Lambda} \leftarrow \boldsymbol{\Lambda}_{MAP}$ ;
3      for $m \in \{1 : M\}$ do
4          for $k \in \{1 : K\}$ do
               // Reparametrization trick
5              Sample $\boldsymbol{\phi}^{(m,k)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ for $k = 1, \cdots, K$ ;
               // Compute stochastic discrepancy terms
6              $\boldsymbol{E}_\tau^{(m,k)} = g_\xi(\boldsymbol{\phi}^{(m,k)}) = \boldsymbol{\mu}_E^{(m)} + \boldsymbol{\sigma}_E^{(m)} \odot \boldsymbol{\phi}^{(m,k)}$ ;
               // Solve discretized RANS equations
7              Solve $\mathcal{R}(\boldsymbol{z}; \boldsymbol{\tau}_\theta(\boldsymbol{u}) + \boldsymbol{W}\boldsymbol{E}_\tau^{(m,k)}) = 0$ to obtain the solution vector $\boldsymbol{z}^{(m,k)}$ ;      // Equation (12)
               // Compute log-likelihoods and their gradients
8              $\ell^{(m,k)}(\theta, \boldsymbol{E}_\tau^{(m,k)})$, $\partial\ell^{(m,k)}/\partial\theta$, $\partial\ell^{(m,k)}/\partial\boldsymbol{E}_\tau^{(m,k)}$ ;      // Equations (B.3), (B.8), (B.9)
9          end
10     end
       // Monte Carlo estimate of ELBO
11     Estimate ELBO $\mathcal{F}$ using Equation (B.10) ;
       // Monte Carlo estimate of the gradient of the ELBO
12     Estimate gradient of ELBO $\nabla_\xi \mathcal{F}$ using Equation (B.12) ;
       // Stochastic Gradient Ascent
13     $\xi^{(n+1)} \leftarrow \xi^{(n)} + \rho^{(n)} \odot \nabla_\xi \mathcal{F}$ ;
14     $n \leftarrow n + 1$ ;
15 end
16 return $\xi$
```

One observes that derivatives of the log-likelihood with respect to $\boldsymbol{\eta}$ are needed. This in turn would imply derivatives of the RANS-model outputs with respect to $\{\theta, \boldsymbol{\Lambda}, \boldsymbol{E}_\tau^{(1:M)}\}$ which appear indirectly through $\boldsymbol{\tau}$. Such derivatives are rendered possible by using an adjoint formulation of the discretized RANS model that yields in effect a *differentiable* solver.

Further details about the derivatives of the ELBO and the use of RANS-model sensitivities obtained by an adjoint formulation can be found in Appendix B. An algorithmic summary of the steps entailed is contained in Algorithm 1. We note finally that the ELBO $\mathcal{F}$ (which serves as a lower bound to the model evidence) and can be used to compare (in a Bayesian sense) alternative models. Such model variations could arise by changing e.g. $\boldsymbol{W}$ i.e. the partition into subdomains which enables the dimensionality reduction of the stochastic model discrepancy vector. One could even employ an adaptive division into subdomains guided by $\mathcal{F}$ so as the data can dictate which regions require more/less refinement.

2.2.5. Predictions

In this section, we describe how *probabilistic*, predictive estimates of any quantity of interest related to the RANS-simulated flow can be produced using the trained model. In particular, one can obtain a *predictive, posterior density* $p(\boldsymbol{z}|\mathcal{D})$ on the whole solution vector $\boldsymbol{z}$ of the RANS equations as follows:

$$\begin{aligned} p(\boldsymbol{z}|\mathcal{D}) &= \int p(\boldsymbol{z}, \boldsymbol{E}_\tau, \theta, \boldsymbol{\Lambda}|\mathcal{D})\, d\boldsymbol{E}_\tau\, d\theta\, d\boldsymbol{\Lambda} \\ &= \int p(\boldsymbol{z}|\boldsymbol{E}_\tau, \theta)\, p(\boldsymbol{E}_\tau, \theta, \boldsymbol{\Lambda}|\mathcal{D})\, d\boldsymbol{E}_\tau\, d\theta\, d\boldsymbol{\Lambda} \\ &= \int p(\boldsymbol{z}|\boldsymbol{E}_\tau, \theta)\, p(\boldsymbol{E}_\tau|\boldsymbol{\Lambda}) p(\theta, \boldsymbol{\Lambda}|\mathcal{D})\, d\boldsymbol{E}_\tau\, d\theta\, d\boldsymbol{\Lambda}. \end{aligned} \tag{32}$$

The third of the densities in the integrand is the posterior which is substituted by its variational approximation i.e. q_ξ in Equation (26) and for the optimal parameter values $\boldsymbol{\xi}$ identified as described in the previous section. The second of the densities represents the prior model prescribed in Equation (20). Finally, the first of the densities is simply a Dirac-delta which corresponds to the solution of the RANS equations obtained when using the proposed closure model for given $\boldsymbol{E}_\tau, \theta$. Since the RANS solver is a black-box, it would not be possible to propagate the uncertainty in $\boldsymbol{E}_\tau$ (and potentially θ) otherwise. In practical terms and given the intractability of this integral, the equation above suggests a Monte Carlo scheme for obtaining samples from $p(\boldsymbol{z}|\boldsymbol{D})$ which involves the following steps. For each sample:

- Set $\theta = \theta_{MAP}$, $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_{MAP}$. (If a different variational approximation to the posterior q_ξ than the one in Equations (27), (28) were used, then $\theta, \boldsymbol{\Lambda}$ would need to be sampled from it).
- Sample $\boldsymbol{E}_\tau$ from $p(\boldsymbol{E}_\tau|\boldsymbol{\Lambda}_{MAP})$ in Equation (20) and compute model discrepancy vector $\boldsymbol{\epsilon}_\tau = \boldsymbol{W}\boldsymbol{E}_\tau$.
- Solve the discretized RANS model in Equation (12) for $\boldsymbol{\tau} = \boldsymbol{\tau}_\theta(\boldsymbol{u}) + \boldsymbol{\epsilon}_\tau$.

Other numerical integration techniques (e.g. Quasi Monte Carlo or Importance Sampling) could be used in order to obtain estimates of the integral with fewer RANS solves. The aforementioned steps would need to be repeated for as many samples as desired.

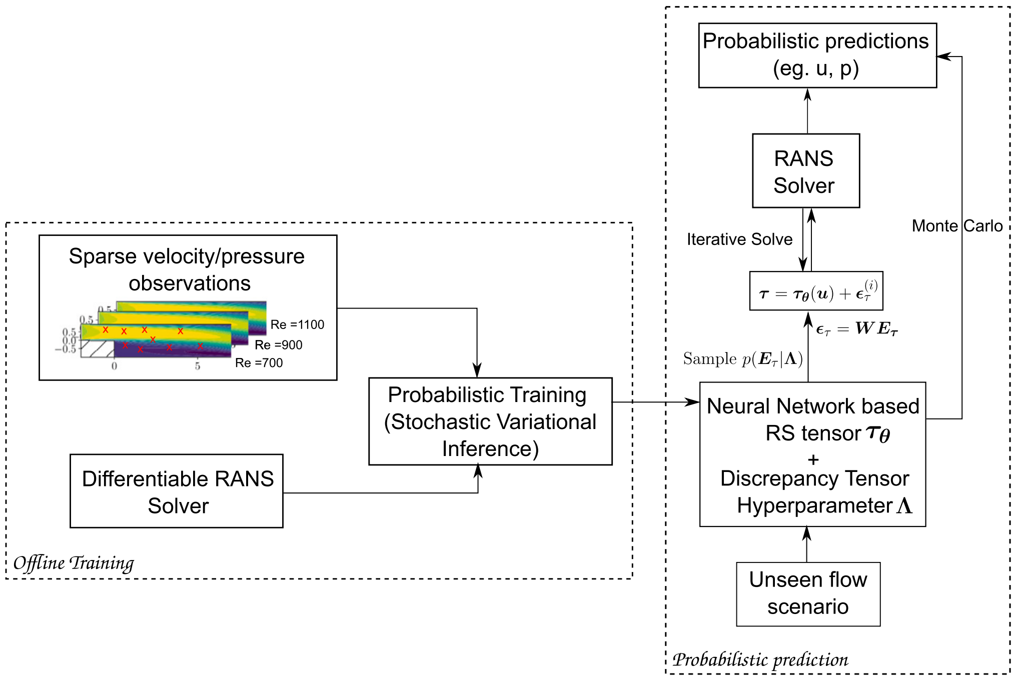

Probabilistic predictions (eg. u, p)
RANS Solver
Iterative Solve
Monte Carlo
$\tau = \tau_\theta(u) + \epsilon_\tau^{(i)}$
$\epsilon_\tau = W E_\tau$
Sample $p(E_\tau|\Lambda)$
Sparse velocity/pressure observations
Re =1100
Re =900
Re =700
Probabilistic Training (Stochastic Variational Inference)
Neural Network based RS tensor τ_θ + Discrepancy Tensor Hyperparameter Λ
Differentiable RANS Solver
Offline Training
Unseen flow scenario
Probabilistic prediction

Fig. 2. Schematic illustration of the training/inference (left block) and probabilistic prediction (right block) framework proposed.

Subsequently, the samples can be used to compute statistics of the predictive estimate (e.g. predictive mean, variance, credible intervals, etc) not only of z (i.e. velocities/pressures) but of any quantity of interest such as the lift, drag, skin friction, etc.

We note however that stochastic RS discrepancy terms ϵ_τ or E_τ and the associated probabilistic model, are limited to the flow geometry used for the training. While it can be used for unseen flow scenarios (e.g. different Re number, inlet conditions, boundary conditions), it cannot be employed for a different flow geometry. In theory, the parametrization of the ϵ_τ can be updated to accommodate different geometries, but we leave it for future investigations. Finally, we would like to highlight the fact that baseline RANS data *is not needed* as an input to the neural networks for prediction in the proposed scheme, as opposed to other frameworks that have been employed in the past (e.g. [22,15,17,65]).

In terms of computational aspects, the stochastic nature of the (reduced) discrepancy tensor E_τ can potentially introduce non-smoothness in the RS vector τ used for solving the RANS equations. In the present work, we have used diagonal covariance for the hyper-prior of E_τ given by $\Lambda = diag(\Lambda^{(J)})$, $J = 1, \dots, N_d$, thus assuming there is no correlation among the nearby nodes/region. As an avenue for future work, a banded covariance matrix can be employed to capture such spatial correlations. Sparsity-inducing priors that account for spatial correlations have been proposed in [80,73]. A schematic overview of the methodological framework discussed in Section 2.2.4 and Section 2.2.5 is presented in Fig. 2. Numerical results, training data and the corresponding source code will be made available at https://github.com/pkmtum/D3C-UQ/ upon publication.

3. Numerical illustrations

3.1. Test case: Backward facing step

We select the backward facing step configuration in order to assess the proposed modeling framework. This is a classic benchmark problem that has been widely used for studying the performance of turbulence models as it poses significant challenges due to the complex flow features such as flow separation, reattachment and recirculation [81,82]. In this setup, as illustrated in Fig. 3 a two-dimensional channel flow is abruptly expanded into a rectangular cavity with a step change in height. The flow separates at the step and forms two recirculation zones downstream, one directly after the step and the other on the upper channel wall downstream. These two recirculation zones affect the reattachment length. The flow features can be seen in more detail as depicted in the LES simulations in Fig. 6. In this setting, the Reynolds number is defined as:

$$Re = \frac{uh}{\nu}, \tag{33}$$

where h and ν are the characteristic length (also the step height) and kinematic viscosity, respectively and u denotes the average velocity of the inlet flow. In the present study, the expansion ratio H/h is 2 and the boundary conditions are shown in Fig. 3. They consist of constant inlet bulk velocity $u_b = 1$ in the x-direction ($u_b = u$) on the left boundary, no-slip condition on Γ_D (i.e., top/bottom boundary) and zero tractions along Γ_N (i.e., at the outflow boundary) i.e. $\left(-pI + \frac{\nu}{2}(\nabla u + \nabla u^T)\right) \cdot \hat{n} = 0$ where $\hat{n}$ is the outward normal of the outflow. On the $x - y$ plane, we place the origin at the corner of the step.

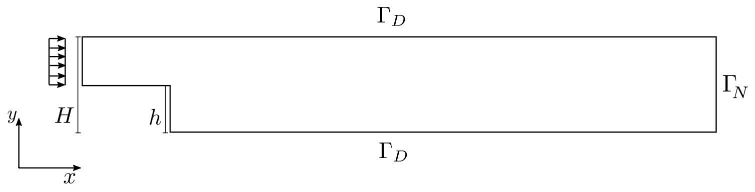


Fig. 3. Backward facing step flow configuration with step height h and the total channel height H. The origin of the $x-y$ plane is placed at the corner of the step. The axes are depicted at the bottom left to avoid clutter.

Table 2
Parameters used for performing the CFD simulations for the generation of the training and test data [51].

Domain size	$27h \times 2h \times h$
DoF LES	1428920
DoF $k-\epsilon$	35709
Step height (h)	1
Total channel height (H)	2
Kinematic viscosity ν values for training data generation ($\times 10^{-3}$)	3.33, 2.00, 1.42, 1.11, 0.909
Re values for training data generation	300, 700, 900, 1100
Re value for prediction	500
Characteristic length	step height h

3.2. Generation of training data

In order to generate the training data, we performed Large Eddy Simulations (LES) for various Reynolds numbers i.e. by varying the kinematic viscosity ν. A three-dimensional configuration is adopted wherein the z-direction (i.e., the in-plane direction) is periodic and the mean fields averaged over the z-direction are used for training. We also performed RANS simulations using the $k-\epsilon$ model for the same set of Re numbers to provide a comparison as it is the most widely used RANS model in industrial applications. In the subsequent discussions, we will refer to the $k-\epsilon$ **model as the *baseline* RANS model**.

We used the open-source CFD platform OpenFOAM [83] for the LES and baseline RANS simulations. We utilized the steady-state, incompressible solver simpleFoam for the baseline RANS simulations. This solver uses the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) in order to solve both the momentum and pressure equations. For the LES simulations, we employed the pimpleFoam transient solver, which combines both the PISO (Pressure Implicit with Split Operator) and SIMPLE algorithms to solve the pressure and momentum equations. In particular, we use the WALE (Wall-Adapting Local Eddy-Viscosity) model [84]. This model is well-suited for capturing turbulent structures near solid walls and is known for its accurate predictions of wall-bounded turbulent flows. In order to overcome the computational demands of LES, a domain decomposition approach was employed. In particular, we split the domain into 16 subdomains and leverage the CPU cores in parallel.[3] We discretized both the baseline RANS and LES domains using second-order methods and all the meshes were non-uniform with mesh density increasing in the domains of interest. To ensure numerical accuracy, we ran all simulations with a CFL number below 0.3. Other pertinent details of the LES and baseline simulations are provided in Table 2. We note that the purpose of the LES model in the present work is to serve as the reference solver that the trained RANS model would try to approximate or match. The framework proposed does not make use of the particulars of the LES solver and one could readily use actual or DNS data (or combinations thereof) for training. The LES data might deviate from physical reality or the results obtained by DNS. Readers interested in the relative accuracy of LES simulations are directed to [85,86].

The mean field reference data from LES is interpolated to the same mesh used for the Finite Element (FE)-based calibrated RANS model (model implementation details discussed in the sequel). However, not all the observations in the grid were used for training. At each RANS grid point, an independent, random coin was flipped with probability 10% of picking the grid point. This resulted in **mean velocity/pressure observations at approximately** 8% **of the grid points to be used as training data**, i.e. a rather sparse dataset (the selected observation points are depicted in Fig. 4). Naturally, the density of RANS grid was higher near the step, so the density of observations was higher in the regions of steeper velocity gradients (in the reattachment and recirculation regions) and lower in other regions. Hence, the training dataset $\mathcal{D} = \{\hat{\boldsymbol{z}}^{(i)}\}_{m=1}^{M}$ consists of the velocities/pressure at those grid points obtained from the $M = 4$ LES simulations carried out with for the corresponding Re numbers (in Table 2). The influence of the number of observation points on the learning and predictive results is an interesting research direction, which we will address in future investigations. A few snapshots of the instantaneous velocity magnitude for $Re = 1100$ from the LES simulation are depicted in Fig. 5 where one can observe how the flow features of interest evolve over time. Only the time-averaged velocities/pressures were used for training which are shown in Fig. 6.

[3] Computations were carried out in a Intel Core i9-12900K CPU.

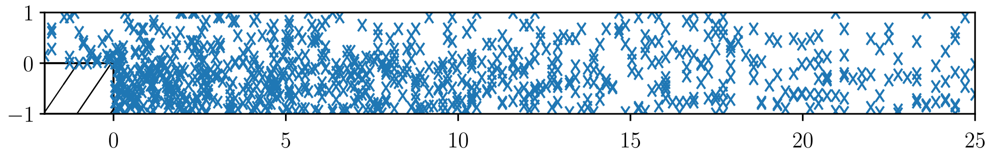


Fig. 4. (Random) grid points where LES velocities/pressure were used for training data. We note that the total number of LES grid points is $\mathcal{O}(10^5)$ whereas LES simulation data at approximately 1000 grid points were employed.

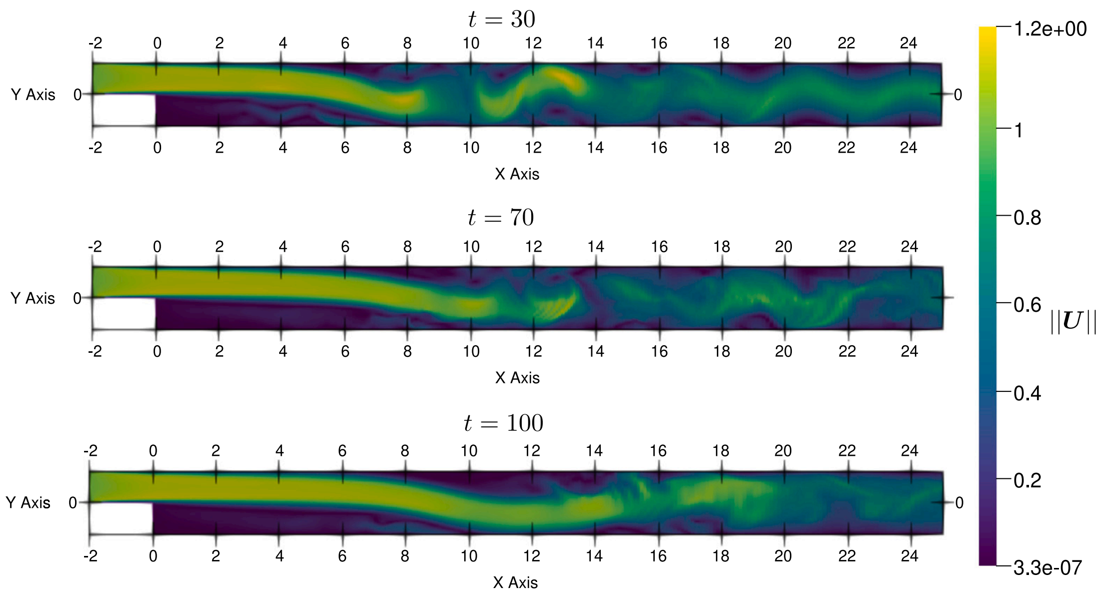


Fig. 5. Instantaneous velocity magnitude $||\boldsymbol{U}||$ at different time-instants $t = \{30, 70, 100\}$ obtained from the LES simulation performed at $Re = 1100$. We note that the flow eventually reaches a stationary state.

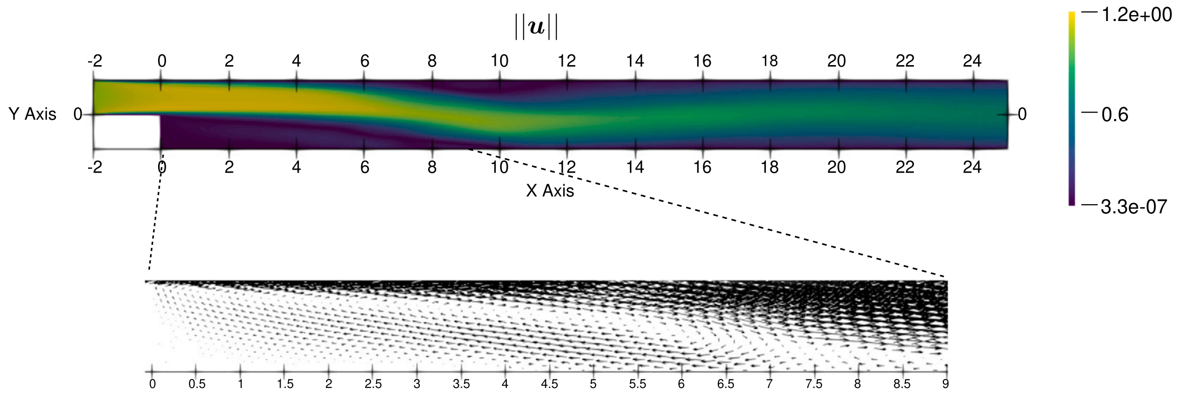


Fig. 6. Time-averaged velocity magnitude $||\boldsymbol{u}||$ obtained from LES simulation performed at $Re = 1100$. The velocity vector plot highlights the flow separation, recirculation zones and the flow reattachment.

3.3. Differentiable forward RANS solver and probabilistic learning implementation

For the discretization of the RANS equations (Equation (12)) the Finite Element (FE) method was employed and the implementation was carried out in the open source package `FEniCS` [87] due to its innate adjoint solver [88]. The basic quantities and their dimensions are listed in Table 3. Further details about the differentiable solver can be found in Appendix A.

Probabilistic inference and learning tasks were performed using the probabilistic programming package `Pyro` [89] which is built on top of the popular machine learning library `PyTorch` [90]. The ELBO maximization was performed using the ADAM scheme [78]. The number of Monte Carlo samples used at each iteration for the estimation of the ELBO and its gradient was $K = 5$ (Algorithm 1). The gradient computation of the ELBO (Equation (31)) was enabled by overloading the `autograd` functionality of PyTorch to facilitate interaction between the solver's adjoint formulation and the auto-differentiation-based neural network gradient. A relatively small learning rate of 10^{-6} was employed due to the Monte Carlo noise in the ELBO gradients. The neural network architecture employed for the parametric RS model was identical to the one suggested by [15] where the optimal number of hidden layers and nodes per layer was determined to be 8 and 30 respectively. The Leaky ReLU was chosen as the activation function for all layers. We noted however that the usual practice of random weight initialization was unsuitable as it led to divergence of the solution

Table 3
Basic quantities and dimensions.

Quantity	dimensions
Domain size	$27h \times 2h$
number of nodes in FE simulation (N)	12180
dim($\boldsymbol{z}$)	12180×3
dim($\boldsymbol{\tau}$)	12180×3
Boolean Matrix dim($\boldsymbol{W}$)	12180×52
dim($\boldsymbol{E}$)	52×3
dim($\boldsymbol{\Lambda}$)	52×3
dim($\boldsymbol{\theta}$)	6970

even after applying the stabilization schemes. For this reason, we used baseline RANS closure data with added noise to pre-train the neural network in order to provide a suitable initialization.

3.4. Results and discussion

We assessed the trained model for the test-case with $Re = 500$ which was not contained in the training data. In the sequel, various aspects of the probabilistic predictions obtained as described in Section 2.2.5 are compared with the reference LES and the baseline RANS predictions. Even though the same *parametrized* RS closure term was used in [15] (and other subsequent works branching from this), their results are not directly comparable due to the use of blending functions [17,22], which combine baseline RS values near the walls with the constant, predicted RS in the bulk and with the amount of blending being case-dependent.

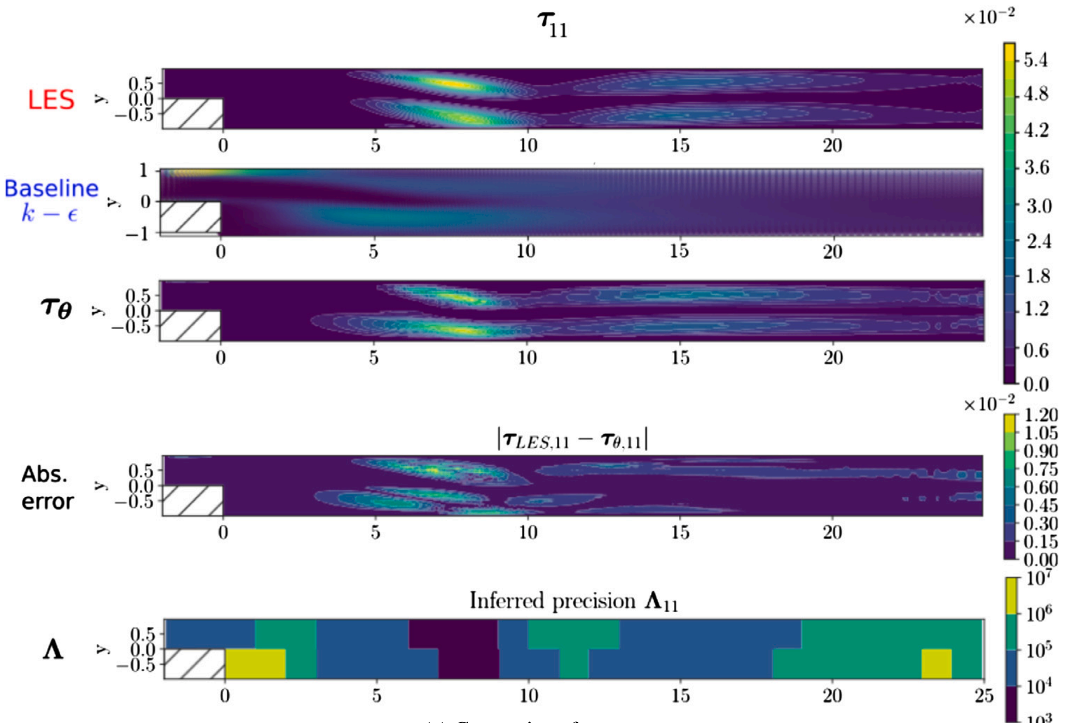
The performance of the proposed method in predicting the components of the RS tensor is shown in Fig. 7, from which the following conclusions can be drawn:

- Even though **no RS observations** were provided during the training, the parametric $\boldsymbol{\tau}_\theta$ (i.e. neural-network based) part of the RS closure is able to capture the basic features of the reference (i.e. LES) RS field. In contrast, the baseline $k - \epsilon$ model severely under/over-estimates its magnitude and misrepresents its spatial variability. We observe that some regions have relatively higher errors (for e.g., around $5 < x/h < 10$ in Fig. 7(c)), which is attributed to the parametric model's inability to provide adequate closure. This model error is exactly what we attempt to capture with our proposed stochastic discrepancy term.
- The bottom row of all the three subfigures in Fig. 7 depicts the inferred precision $\boldsymbol{\Lambda}$ of the (reduced) discrepancy tensor $\boldsymbol{E}_\tau$. As this is inversely proportional to the predictive uncertainty we note that it attains smaller values in the regions where the parametric model $\boldsymbol{\tau}_\theta$ deviates the most from the LES values (e.g. for $5 < x/h < 10$). Conversely, it attains very large values (which correspond to practically zero model discrepancies) in areas where the parametric closure model is able to correctly account for the underlying phenomena (e.g. far downstream and for all three RS components). This is expected as the flow attains an almost parabolic profile in this region, which in turn translates to reduced fluctuations in the RS tensor, making it easier for the neural network to learn.

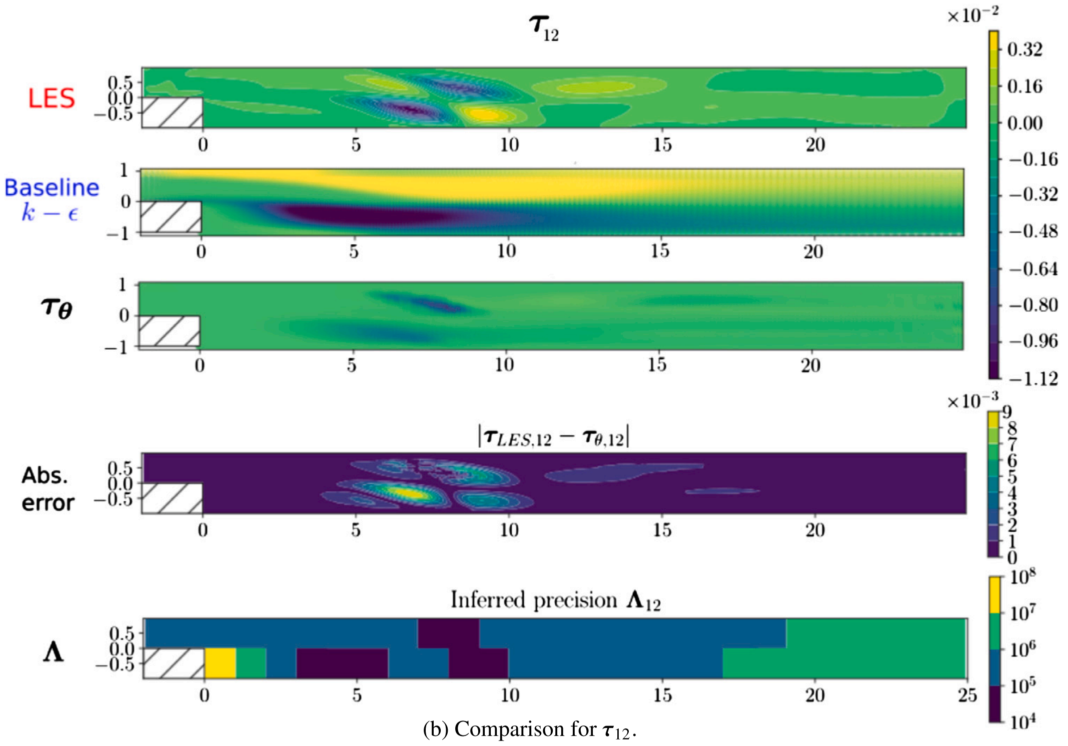
The Monte-Carlo-based scheme (detailed in Section 2.2.5) was employed to propagate the model form uncertainty forward in order to obtain probabilistic predictive estimates for the quantities of interest i.e. mean stream-wise velocity u, wall-normal velocity v (Fig. 8, Fig. 9) and mean pressure p (Fig. 10). Cross-sections of the aforementioned quantities are depicted in Fig. 11. The following conclusions can be drawn from the Figures:

- Even though the RS field is captured with some discrepancies, the predicted mean fields agree well with the reference LES data (Figs. 8, 9 and 10, discussed in the sequel). This points to the non-uniqueness of this inverse problem solution, also reported by other works [7,37].
- There exist two recirculation zones in the backward-facing step flow setup. The primary one forms just after the step and the secondary appears above it for Reynolds numbers close to 400 and above, for the given expansion ratio [91]. As it can be seen in Fig. 8, the proposed model is able to predict the appearance of the two recirculation zones in close agreement with the LES, whereas the baseline RANS model underestimates the size of the first recirculation zone and almost completely misses the second one.
- The last row of Fig. 8 depicts the predictive posterior standard deviation of the aforementioned quantities. As expected, around the shear layers (the top of the first recirculation zone and the bottom of the second recirculation zone), the uncertainty is the highest. This is even more clearly observed in the cross-sections of Fig. 11 which illustrate the predictive, posterior mean plus/minus two posterior standard deviations. More importantly perhaps, one observes that the predictions envelop the reference LES values in most areas. The model is extremely confident close to the inlet, as manifested by the very tight credible interval. As one moves further downstream and close to the first recirculation zone, the parametric closure suffers, hence the uncertainty bounds are wider to account for it. The ability to quantify aleatoric, predictive uncertainty[4] is one of the main advantages of the

[4] As mentioned earlier, MAP point-estimates for the model parameters θ were used.



(a) Comparison for τ_{11}.

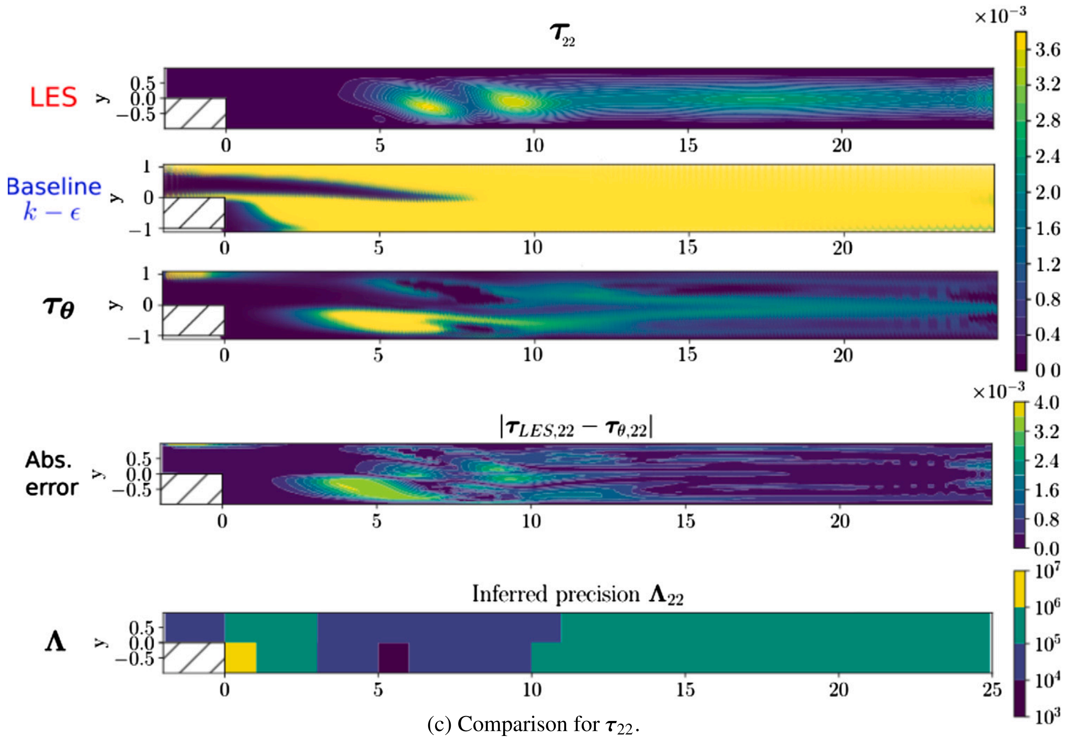


(b) Comparison for τ_{12}.

Fig. 7. Comparison of the predicted components of the RS tensor τ_θ with the LES reference values and the *baseline* RANS ($k-\epsilon$) for the test case with $Re = 500$. The three components τ_{11}, τ_{12} and τ_{22} are separately compared in subfigures (a), (b), and (c) respectively (*subfigure (c) in the next page*). In each block, **top** - the LES RS tensor component contour, **second** - the $k-\epsilon$ RS tensor component contour, **third** - the predicted, parametric RS tensor τ_θ for $\theta = \theta_{MAP}$, **fourth** - the contour plot of the absolute error between the LES RS tensor and the τ_θ tensor component, **bottom** - the inferred hyper-parameter Λ of the (reduced) discrepancy tensor E_τ.

probabilistic model proposed in contrast to the more commonly used deterministic counterparts as well as alternatives that can only capture epistemic uncertainty.

- Similarly to the stream-wise velocity, predictions for the wall-normal velocity (Fig. 9) and the pressure (Fig. 10) are in good agreement with the reference LES values, as opposed to the baseline RANS. In the first recirculation zone, the baseline $k-\epsilon$ is completely off, while the predicted values with the credible interval cover the reference LES. Also, the pressure predictions (Fig. 10) identify the crucial zone where the flow reattaches to the wall (around $x/h \approx 10$), which is very difficult to predict in general. At the reattachment point, there is a transition from low-pressure in the recirculation zone to higher pressure along the



(c) Comparison for τ_{22}.

Fig. 7. (*continued*)

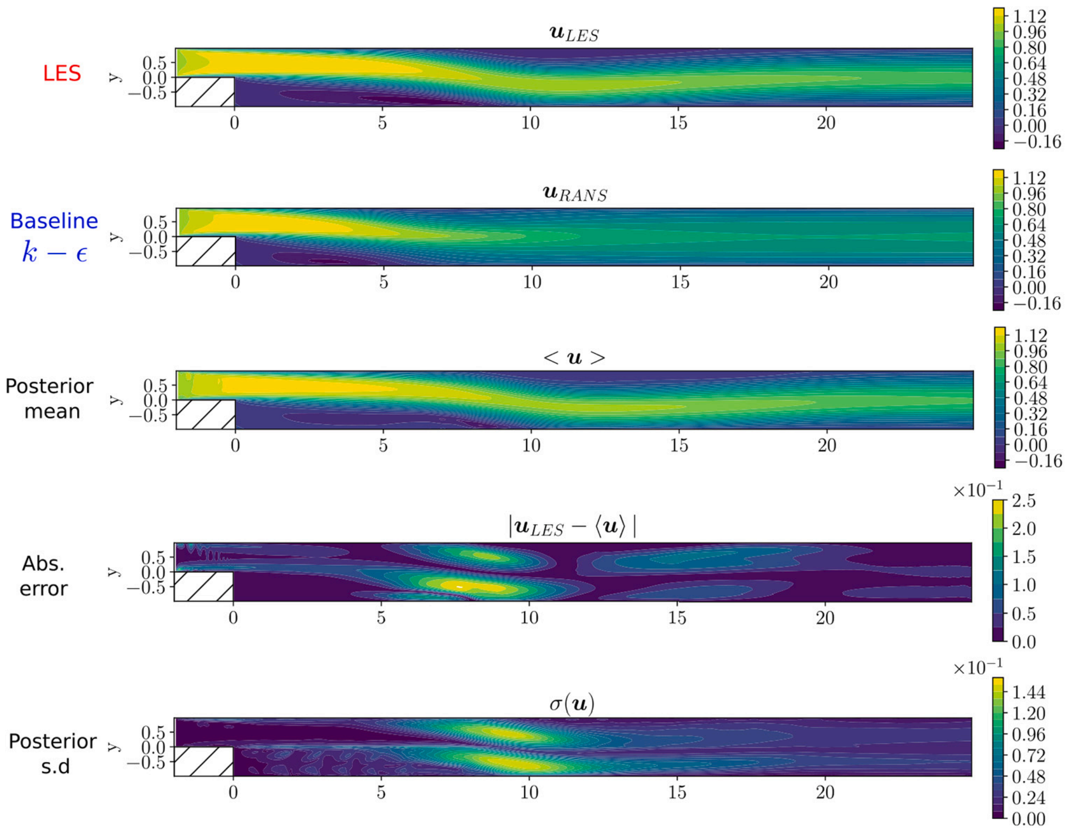


Fig. 8. Velocity contours in x-direction ($\boldsymbol{u}$) for the test case with Re = 500. **top** - ground truth (LES), **second** - baseline $k-\epsilon$, **third** - the posterior predictive mean ($\langle \boldsymbol{u} \rangle$), **fourth** - the contour plot of the absolute error between the LES ($\boldsymbol{u}_{LES}$) and the posterior predictive mean ($\langle \boldsymbol{u} \rangle$), **bottom** - the standard deviation of the posterior predictive ($\sigma(\boldsymbol{u})$).

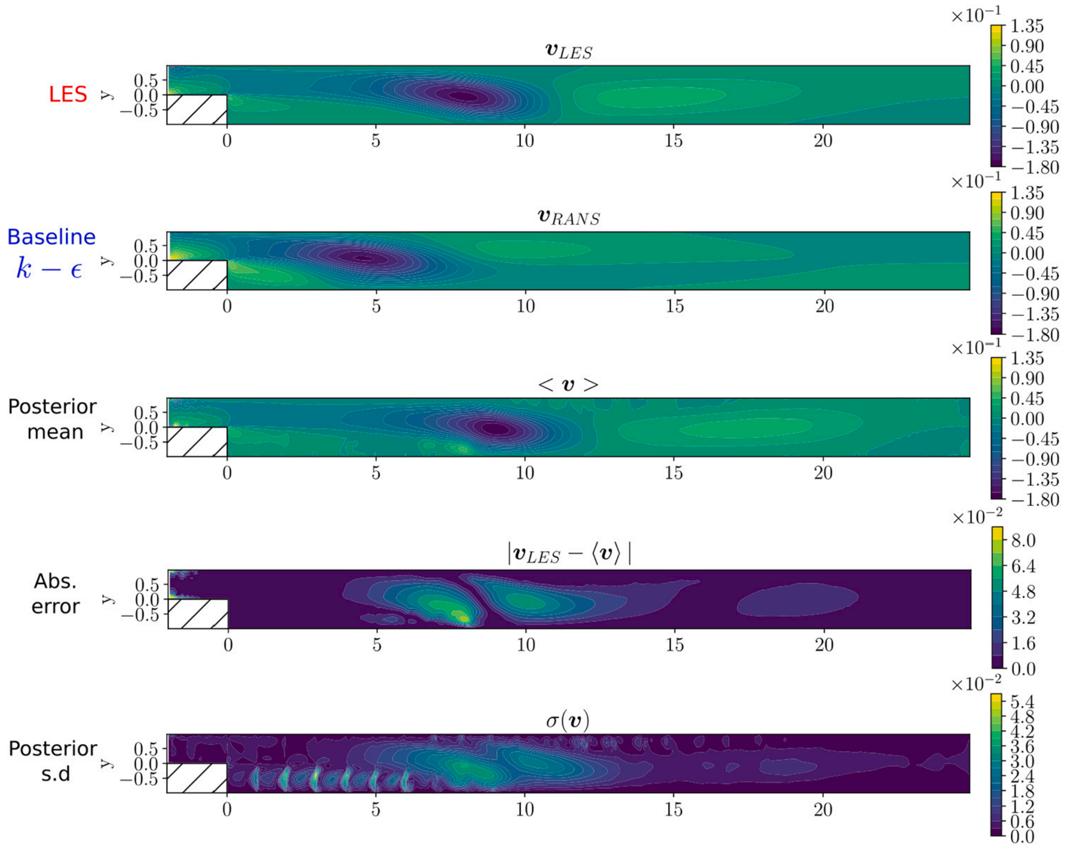


Fig. 9. Velocity contours in y-direction ($\boldsymbol{v}$) for the test case with Re = 500. **top** - ground truth (LES), **second** - baseline $k-\epsilon$, **third** - the posterior predictive mean ($\langle\boldsymbol{v}\rangle$), **fourth** - the contour plot of the absolute error between the LES ($\boldsymbol{v}_{LES}$) and the posterior predictive mean ($\langle\boldsymbol{v}\rangle$), **bottom** - the standard deviation of the posterior predictive ($\sigma(\boldsymbol{v})$).

Table 4
Relative computational cost in terms of computational time (on an Intel Core i9-12900K CPU) of a single model run for LES and of the proposed, differentiable RANS solver. These are compared with the cost of the baseline RANS model for Reynolds number 500.

	LES	proposed RANS solver
Relative cost	7679	1.3

wall. The posterior standard deviation at this point is also higher than in the other regions, ensuring that the reference solution is enveloped.

As previously mentioned, observations of mean velocities/pressures at approximately 8% of the total number of grid points in the FE mesh were used for training. Fig. 12 highlights this by comparing the stream-wise velocity at three different sections x/h. The left subfigure depicts the section in the first re-circulation zone. It can be seen that despite having very few observation points near the wall, the u predictions are able to capture the backward flow in the re-circulation regions. In contrast, the baseline $k-\epsilon$ completely fails to capture it. This can be attributed to the earlier reattachment of the flow in the baseline $k-\epsilon$ case (flow reattachment discussed in the sequel). The middle and the right subfigures depict sections further downstream, with the middle being in the second re-circulation zone and the right in the flow recovery zone. It is observed that with a relatively small number of observation points, the trained model's prediction is in agreement with the LES. Furthermore, the latter is enveloped by the credible interval constructed by considering $\pm 3\times$ (the posterior standard deviation). This credible interval is much tighter as compared to the one in the left subfigure.

The primary motivation behind RANS models and of this work is to reduce the computational cost of flow simulations, especially in cases where LES or DNS are prohibitively expensive. In Table 4 we report the computational time of a single, model run as compared to the baseline RANS model (for $Re = 500$). We observe that even though the proposed differentiable solver is of comparable cost to the baseline RANS, it can provide far more accurate, probabilistic predictions of the LES model's outputs which is roughly 6000 times more computationally demanding.

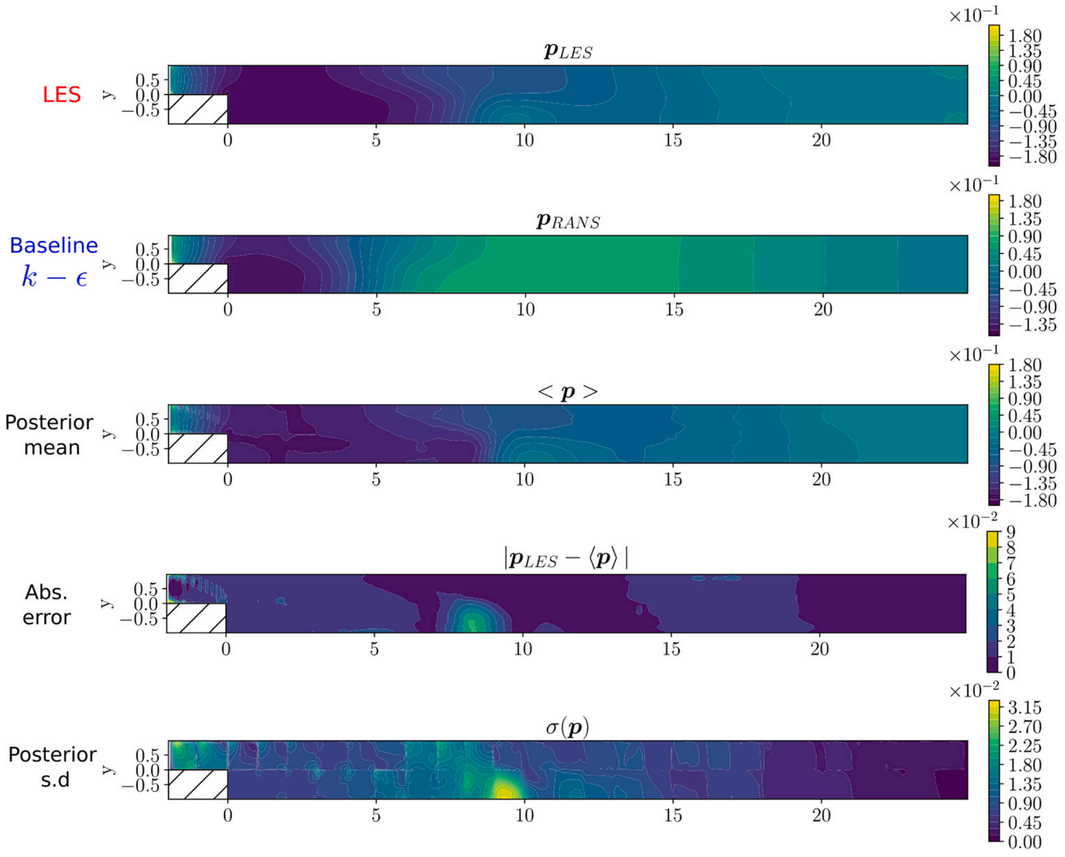


Fig. 10. Pressure contours (p) for the test case with Re $=$ 500. **top** - ground truth (LES), **second** - baseline $k-\epsilon$, **third** - the posterior predictive mean ($\langle \boldsymbol{u} \rangle$), **fourth** - the contour plot of the absolute error between the LES (p_{LES}) and the posterior predictive mean ($\langle p \rangle$), **bottom** - the standard deviation of the posterior predictive ($\sigma(\boldsymbol{u})$).

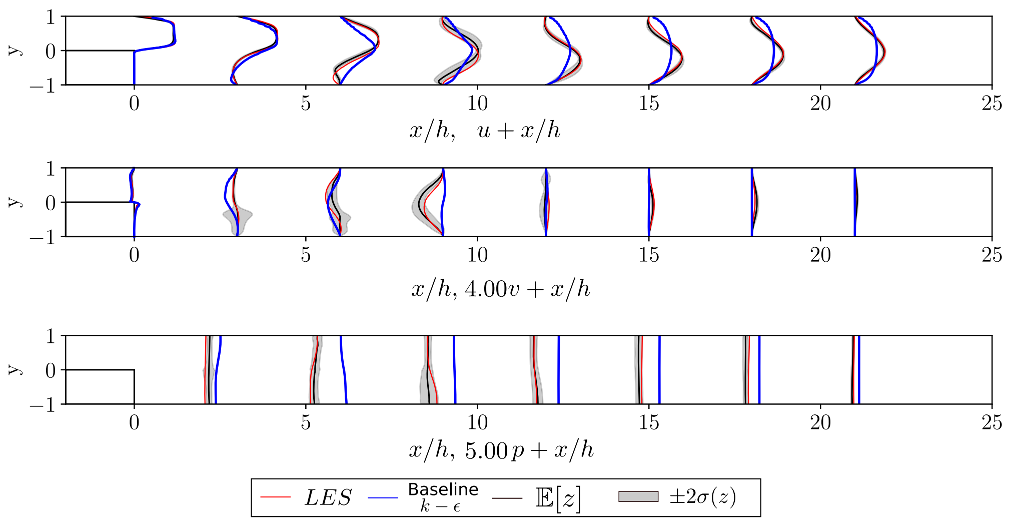


Fig. 11. Section plots at different locations x/h comparing the LES and *Baseline* RANS mean fields with the posterior predictive mean (solid black line) and $\pm 2\times$ standard deviation (shaded area) for the test case with $Re = 500$. **top** - velocity in x-direction ($\boldsymbol{u}$), **middle** - velocity in the y-direction ($\boldsymbol{v}$), **bottom** - pressure ($\boldsymbol{p}$). $\mathbb{E}[z]$ and $\sigma(z)$ depict the predictive posterior mean and standard deviation respectively.

Accurately capturing the recirculation zones is crucial to getting a reliable estimate of the reattachment length, which is a key parameter in the study of separated flows, such as the case here. The reattachment length is defined as the distance from the step where the flow separates to the point at which it reattaches to the surface downstream of the step. Reattachment occurs where the velocity gradient off the wall is zero, or in other words, where the wall shear stress is zero. The predicted reattachment length ($x_{reattach}$) by the proposed method is compared with a) the LES data (Section 3.2) b) the baseline RANS (Section 3.2) c) the two-dimensional, LES simulation performed by [92] for the expansion ratio of 1.9423, and d) the results in [22], who used the same Tensor Basis Neural Networks (TBNN) [15] employed in the parametric closure term in our work as well. The results are summarized

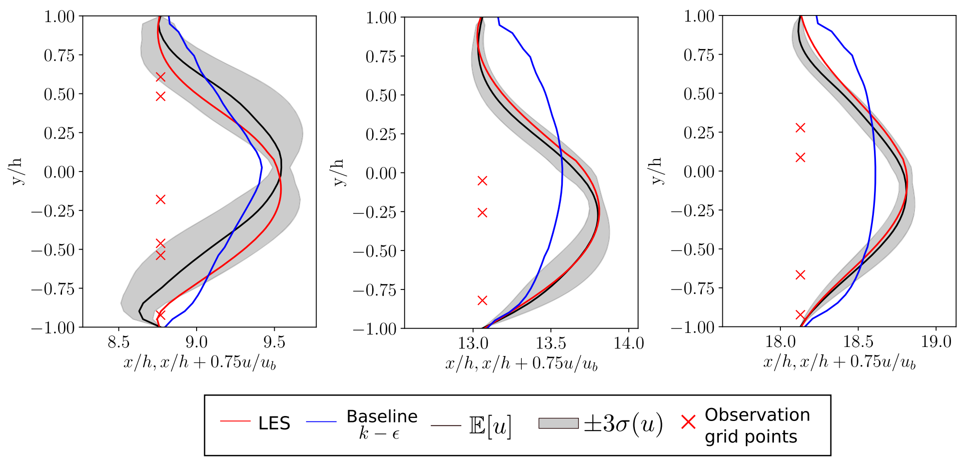


Fig. 12. Section plots of the stream-wise velocity u at three different x/h locations for the test case with $Re = 500$. In addition, the points where training observations were available are depicted with red crosses. $\mathbb{E}[u]$ and $\sigma(u)$ depict the predictive posterior mean and standard deviation respectively.

Table 5
Predictions of reattachment length (x/h) of the primary recirculation region behind the backward-facing step (expansion ratio $H/h = 2$) for the test case with $Re = 500$. The $\pm$ corresponds to $3\times$ the posterior standard deviation.

Model	$x_{reattach}$ [x/h]
LES (reference)	9.10
Biswas et al. [92] ($H/h = 1.9423$)	8.9
Baseline RANS ($k-\epsilon$ model)	5.61
Geneva et al. [22]	5.52
proposed model	10.06 ± 1.21

in Table 5 where it is evident that while previous works deviated significantly from the reference LES value, our probabilistic prediction is able to envelop it. The reattachment length is heavily dependent on correctly identifying the two recirculation zones and the baseline RANS model fails to predict the secondary recirculation zone (Fig. 8). This might have resulted in such a low reattachment length. The estimate of the reattachment length is also low in [22], which could be attributed to the lack of the stochastic, model correction term.

4. Conclusions

We have presented a data-driven model for RANS simulations that quantifies and propagates in its predictions an often neglected source of uncertainty, namely the aleatoric, model uncertainty in the closure equations. We have combined this with a parametric closure model which employs a set of tensor basis functions that depend on the invariants of the rate of strain and rotation tensors. A fully Bayesian formulation is advocated which makes use of a sparsity-inducing prior in order to identify the regions in the problem domain where the parametric closure is insufficient and in order to quantify the stochastic correction to the Reynolds stress tensor. We have demonstrated how the model can be trained using sparse, indirect data, namely mean velocities/pressures in contrast to the majority of pertinent efforts that require direct, RS data. While the training data in our illustrations arose from a higher-fidelity model, one can readily envision using experimental observations as well.

In order to enable inference and learning tasks, we developed a differentiable RANS solver capable of providing parametric sensitivities. Such a differentiable solver was non-trivial owing to the complexity of the physical simulator and its stability issues. The lack of such numerical tools has proven to be a significant barrier for intrusive, physics-based, data-driven models in turbulence [26]. This differentiable solver was utilized in the context of a Stochastic Variational Inference (SVI) scheme that employs Monte Carlo estimates of the ELBO derivatives in conjunction with the reparametrization trick and stochastic gradient ascent. We demonstrated how probabilistic predictive estimates can be computed for all output quantities of the trained RANS model and illustrated their accuracy on a separated flow in the backward facing step benchmark problem. In most cases, very good agreement with the reference values was achieved and in all cases these were enveloped by the credible intervals computed.

The proposed modeling framework offers several possibilities for extensions, some of which we discuss below:

- The indirect data i.e. velocities/pressures as in Equation (22), could be complemented with direct, RS data at certain locations of the problem domain. This could be beneficial in improving the model's predictive accuracy and generalization capabilities.
- The parametric closure model could benefit from non-local dependencies which could be enabled by convolutional or vector-cloud neural networks (VCNN) [93] with appropriate embedding of invariance properties.

- The dimensionality reduction of the stochastic discrepancy terms (Equation (19)) was based on a pre-selected and uniform division of the problem domain into subdomains. The accuracy of the model would certainly benefit from a learnable and adaptive scheme that would be able to focus on the areas where model deficiencies are most pronounced and stochastic corrections are most needed.

CRediT authorship contribution statement

Atul Agrawal: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Phaedon-Stelios Koutsourelakis:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

A link to a github repository has been provided and pertinent code and data will be made available upon publication

Acknowledgements

We extend our sincere thanks to Jigar Parekh for his valuable guidance in the generation of the LES data and to Maximilian Rixner for his assistance in implementing the forward solver. We would also like to thank Nicholas Geneva and Niklas Fehn for the valuable discussions that enriched this work. We finally extend sincere gratitude to the anonymous reviewers, whose insightful feedback greatly enhanced the quality of this manuscript.

Appendix A. Differentiable RANS solver

In the present study, the RANS equations (Equation (7)) are numerically solved using the finite element discretization, implemented in the open source package FEniCS [87]. The discrete equations are obtained by representing the solution and test functions in appropriate finite dimensional function spaces. In particular, we employed the standard Taylor-Hood pair of basis functions with polynomial degree one for the pressure interpolants and two for the velocities. This choice is made to avoid stability issues potentially arising from the interaction between the momentum and continuity equations.

The turbulence scaling terms, k and ϵ in Equation (17), are obtained by solving the respective standard transport equations [2,52]. Symmetry is enforced in the RS tensor, i.e. τ_{xy} and τ_{yx} are identical without any redundancy in the representation. The discretized system is solved with damped Newton's method. For robustness and global convergence, pseudo-time stepping is used with the backward Euler discretization [94]. As the Reynolds number is increased, the convection term dominates, leading to stability [95]. This elicits a need to add stabilization terms to the weak form, such as the least-square stabilization, according to which the weighted square of the strong form is added to the weak form residual. However, these extra terms have to be chosen carefully in order not to compromise the correctness of the approximate solution. Classically, researchers added artificial diffusion terms or a numerical diffusion by using upwind scheme for the convection term instead of central diffusion. The extra infused term corrupted the solution quality. To avoid this, in practice, it is common to use schemes like Streamline-Upwind Petrov-Galerkin method (SUPG) and Galerkin Least Squares (GLS). In the present study, we have utilized a self-adjoint numerical stabilization scheme which is an extension of Galerkin Least Squares (GLS) Stabilization called Galerkin gradient least square method [96]. This amounts to adding a stabilization term to the residual weak form. For additional details, interested readers are referred to [96,95].

Appendix B. Adjoint formulation and estimation of the gradient of the ELBO

As discussed in Section 2.2.4, the SVI framework advocated, in combination with the reparametrization trick, requires derivatives of the ELBO with respect to the variables which we summarily denoted by $\boldsymbol{\eta} = \{\boldsymbol{\theta}, \boldsymbol{\Lambda}, \boldsymbol{E}_\tau^{(1:M)}\}$, i.e. (as in Equation (31)):

$$\nabla_\xi \mathcal{F}(\xi) = \mathbb{E}_{q(\boldsymbol{\phi})} \left[\nabla_\xi g_\xi(\boldsymbol{\phi}) \nabla_{\boldsymbol{\eta}} \left(\log p(\mathcal{D}, \boldsymbol{\eta}) - \log q_\xi(\boldsymbol{\eta}) \right) \right], \tag{B.1}$$

where from Equation (23):

$$\begin{aligned} \log p(\mathcal{D}, \boldsymbol{\eta}) &= \log \left(p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{E}_\tau^{(1:M)})\ p(\boldsymbol{E}_\tau^{(1:M)} | \boldsymbol{\Lambda})\ p(\boldsymbol{\theta})\ p(\boldsymbol{\Lambda}) \right) \\ &= \left(\sum_{m=1}^M \log p(\hat{\boldsymbol{z}}^{(m)} \mid \boldsymbol{\theta}, \boldsymbol{E}_\tau^{(m)}) + \log p(\boldsymbol{E}_\tau^{(m)} | \boldsymbol{\Lambda}) \right) + \log p(\boldsymbol{\theta}) + \log\ p(\boldsymbol{\Lambda}). \end{aligned} \tag{B.2}$$

The form of the (log-)priors $p(\boldsymbol{E}_\tau^{(m)}|\boldsymbol{\Lambda})$ (Equation (20)), $p(\boldsymbol{\theta})$ (Equation (18)), $p(\boldsymbol{\Lambda})$ (Equation (21)) as well as of the approximate posterior $q_\xi(\boldsymbol{\eta})$ (Equations (26) and (27), (28), (29)) suggest that most of these derivatives can be analytically computed with the exception of the ones involving the log-likelihoods, i.e.:

$$\ell^{(m)}(\boldsymbol{\theta}, \boldsymbol{E}_\tau^{(m)}) = \log p(\hat{\boldsymbol{z}}^{(m)} \mid \boldsymbol{\theta}, \boldsymbol{E}_\tau^{(m)}). \tag{B.3}$$

This is because each of these terms depends implicitly on $\boldsymbol{\theta}, \boldsymbol{E}_\tau^{(m)}$ through the output of the RANS solver $\boldsymbol{z}(\boldsymbol{\theta}, \boldsymbol{\epsilon}_\tau^{(m)} = \boldsymbol{W}\boldsymbol{E}_\tau^{(m)})$ with the closure model for the discretized RS tensor field suggested by Equation (13) i.e. $\boldsymbol{\tau} = \boldsymbol{\tau}_{\boldsymbol{\theta}}(\boldsymbol{u}) + \boldsymbol{W}\boldsymbol{E}_\tau^{(m)}$. In view of the governing equations (Equation (12)), we explain below how adjoint equations can be formulated that enable efficient computation of the aforementioned derivatives of the log-likelihoods.

In particular, and if we drop the superscript m for each term in the log-likelihood in order to simplify the notation, we formulate a Lagrangian with the help of a vector λ of Lagrangian multipliers, i.e.:

$$\mathcal{L} = \ell + \lambda^T(\mathcal{G}(\boldsymbol{z}) - \boldsymbol{B}\boldsymbol{\tau}), \tag{B.4}$$

where $\mathcal{G}$, $\boldsymbol{B}$, $\boldsymbol{\tau}$ and $\boldsymbol{z}$ are as defined in Section 2.2. Differentiating with respect to $\boldsymbol{\tau}$ yields:

$$\begin{aligned}\frac{d\mathcal{L}}{d\boldsymbol{\tau}} &= \frac{\partial \ell}{\partial \boldsymbol{z}}\frac{d\boldsymbol{z}}{d\boldsymbol{\tau}} + \frac{d\lambda^T}{d\boldsymbol{\tau}}(\mathcal{G}(\boldsymbol{z}) - \boldsymbol{B}\boldsymbol{\tau}) + \lambda^T\left(\frac{\partial \mathcal{G}}{\partial \boldsymbol{z}}\frac{d\boldsymbol{z}}{d\boldsymbol{\tau}} - \boldsymbol{B}\right)\\ &= \left(\frac{\partial \ell}{\partial \boldsymbol{z}} + \lambda^T\frac{\partial \mathcal{G}}{\partial \boldsymbol{z}}\right)\frac{d\boldsymbol{z}}{d\boldsymbol{\tau}} - \lambda^T\boldsymbol{B}.\end{aligned} \tag{B.5}$$

We select λ^T so that the first term in parentheses vanishes, i.e.:

$$\frac{\partial \ell}{\partial \boldsymbol{z}} + \lambda^T\frac{\partial \mathcal{G}}{\partial \boldsymbol{z}} = 0 \quad \text{or,} \quad \left(\frac{\partial \mathcal{G}}{\partial \boldsymbol{z}}\right)^T \lambda = -\left(\frac{\partial \ell}{\partial \boldsymbol{z}}\right)^T. \tag{B.6}$$

The linear system of equations was solved using a direct LU solver. The vector λ found was substituted in Equation (B.5) in order to obtain the desired gradient which is given by:

$$\frac{d\mathcal{L}}{d\boldsymbol{\tau}} = \frac{d\ell}{d\boldsymbol{\tau}} = -\lambda^T\boldsymbol{B}. \tag{B.7}$$

Subsequently, and by application of the chain rule we can obtain derivatives with respect to $\boldsymbol{\theta}$ as:

$$\frac{d\ell}{d\boldsymbol{\theta}} = \underbrace{\frac{\partial \ell}{\partial \boldsymbol{\tau}}}_{\substack{\text{Adjoint}\\ \text{model}}}\underbrace{\frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\theta}}}_{\substack{\text{NN}\\ \text{auto-diff}}}, \tag{B.8}$$

where $\partial\boldsymbol{\tau}/\partial\boldsymbol{\theta}$ was efficiently computed by back-propagation, which is a reverse accumulation automatic differentiation algorithm for deep neural networks that applies the chain rule on a per-layer basis. We note that since the parameters $\boldsymbol{\theta}$ are common for each likelihood $\ell^{(m)}$ the aforementioned terms would need to be added as per Equation (B.2).

Similarly by chain rule, the gradient with respect to the vector $\boldsymbol{E}_\tau$ is given by:

$$\frac{d\ell}{d\boldsymbol{E}_\tau} = \boldsymbol{W}^T\frac{d\ell}{d\boldsymbol{\tau}}. \tag{B.9}$$

We note finally that the expectations involved in the ELBO and its gradient (Equation (31)) are approximated by Monte Carlo i.e.:

$$\begin{aligned}\mathcal{F}(\boldsymbol{\xi}) \approx \frac{1}{K}\left(\sum_{k=1}^{K}\left(\sum_{m=1}^{M}\ell^{(m)}(\boldsymbol{\theta}, \boldsymbol{E}_\tau^{(m,k)}) + \log p(\boldsymbol{E}_\tau^{(m,k)}|\boldsymbol{\Lambda})\right)\right.\\ \left. + \log p(\boldsymbol{\theta}) + \log p(\boldsymbol{\Lambda}) - \log q_{\boldsymbol{\xi}}(\boldsymbol{\theta}, \boldsymbol{\Lambda}, \boldsymbol{E}_\tau^{(m,k)})\right),\end{aligned} \tag{B.10}$$

where:

$$\boldsymbol{\phi}^{(m,k)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \quad \boldsymbol{E}_\tau^{(m,k)} = g_{\boldsymbol{\xi}}(\boldsymbol{\phi}^{(m,k)}), \tag{B.11}$$

and:

$$\nabla_{\boldsymbol{\xi}}\mathcal{F}(\boldsymbol{\xi}) \approx \frac{1}{K}\sum_{k=1}^{K}\nabla_{\boldsymbol{\xi}}g_{\boldsymbol{\xi}}(\boldsymbol{\phi}^{(k)})\nabla_{\boldsymbol{\eta}}\left(\log p(\mathcal{D}, \boldsymbol{\eta}^{(k)}) - \log q_{\boldsymbol{\xi}}(\boldsymbol{\eta}^{(k)})\right), \tag{B.12}$$

where $\boldsymbol{\eta}^{(k)} = g_{\boldsymbol{\xi}}(\boldsymbol{\phi}^{(k)})$.

References

[1] I. Marusic, S. Broomhall, Leonardo da Vinci and fluid mechanics, Annu. Rev. Fluid Mech. 53 (1) (2021) 1–25, https://doi.org/10.1146/annurev-fluid-022620-122816.
[2] S.B. Pope, Turbulent Flows, Cambridge University Press, 2000.
[3] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences, NNASA/CR-2014-218178, 2014.
[4] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. 52 (2020) 477–508.

[5] R. Vinuesa, S.L. Brunton, Emerging trends in machine learning for computational fluid dynamics, Comput. Sci. Eng. 24 (5) (2022) 33–41.

[6] D. Lucor, A. Agrawal, A. Sergent, Simple computational strategies for more effective physics-informed neural networks modeling of turbulent natural convection, J. Comput. Phys. 456 (2022) 111022, https://doi.org/10.1016/j.jcp.2022.111022.

[7] K. Duraisamy, Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence, Phys. Rev. Fluids 6 (5) (2021) 050504, https://doi.org/10.1103/PhysRevFluids.6.050504, arXiv:2009.10675 [physics].

[8] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence modeling in the age of data, Annu. Rev. Fluid Mech. 51 (2019) 357–377.

[9] T.A. Oliver, R.D. Moser, Bayesian uncertainty quantification applied to RANS turbulence models, J. Phys. Conf. Ser. 318 (2011) 042032, https://doi.org/10.1088/1742-6596/318/4/042032.

[10] E.J. Parish, K. Duraisamy, A paradigm for data-driven predictive modeling using field inversion and machine learning, J. Comput. Phys. 305 (2016) 758–774.

[11] A.P. Singh, S. Medida, K. Duraisamy, Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils, AIAA J. 55 (7) (2017) 2215–2227.

[12] B.D. Tracey, K. Duraisamy, J.J. Alonso, A machine learning strategy to assist turbulence model development, in: 53rd AIAA Aerospace Sciences Meeting, 2015, p. 1287.

[13] H. Xiao, J.-L. Wu, J.-X. Wang, R. Sun, C. Roy, Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach, J. Comput. Phys. 324 (2016) 115–136.

[14] Z.J. Zhang, K. Duraisamy, Machine learning methods for data-driven turbulence modeling, in: 22nd AIAA computational fluid dynamics conference, 2015, p. 2460.

[15] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. 807 (2016) 155–166.

[16] J. Ling, R. Jones, J. Templeton, Machine learning strategies for systems with invariance properties, J. Comput. Phys. 318 (2016) 22–35, https://doi.org/10.1016/j.jcp.2016.05.003.

[17] M.L. Kaandorp, R.P. Dwight, Data-driven modelling of the Reynolds stress tensor using random forests with invariance, Comput. Fluids 202 (2020) 104497.

[18] J.-X. Wang, J. Wu, J. Ling, G. Iaccarino, H. Xiao, A comprehensive physics-informed machine learning framework for predictive turbulence modeling, arXiv preprint, arXiv:1701.07102, 2017.

[19] M. Schmelzer, R.P. Dwight, P. Cinnella, Discovery of algebraic Reynolds-stress models using sparse symbolic regression, Flow Turbul. Combust. 104 (2020) 579–603.

[20] X.-L. Zhang, H. Xiao, X. Luo, G. He, Ensemble Kalman method for learning turbulence models from indirect observation data, J. Fluid Mech. 949 (2022) A26, https://doi.org/10.1017/jfm.2022.744.

[21] S.B. Pope, A more general effective-viscosity hypothesis, J. Fluid Mech. 72 (2) (1975) 331–340.

[22] N. Geneva, N. Zabaras, Quantifying model form uncertainty in Reynolds-averaged turbulence models with Bayesian deep neural networks, J. Comput. Phys. 383 (2019) 125–147.

[23] S. Taghizadeh, F.D. Witherden, S.S. Girimaji, Turbulence closure modeling with data-driven techniques: physical compatibility and consistency considerations, New J. Phys. 22 (9) (2020) 093023.

[24] R.L. Thompson, L.E.B. Sampaio, F.A. de Bragança Alves, L. Thais, G. Mompean, A methodology to evaluate statistical errors in DNS data of plane channel flows, Comput. Fluids 130 (2016) 1–7.

[25] J. Wu, H. Xiao, R. Sun, Q. Wang, Reynolds-averaged Navier–stokes equations with explicit data-driven Reynolds stress closure can be ill-conditioned, J. Fluid Mech. 869 (2019) 553–586.

[26] K. Cranmer, J. Brehmer, G. Louppe, The frontier of simulation-based inference, Proc. Natl. Acad. Sci. USA 117 (48) (2020) 30055–30062.

[27] D.A. Bezgin, A.B. Buhendwa, N.A. Adams, A fully-differentiable compressible high-order computational fluid dynamics solver, arXiv preprint, arXiv:2112.04979, 2021.

[28] B. List, L.-W. Chen, N. Thuerey, Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons, J. Fluid Mech. 949 (2022) A25.

[29] K. Um, R. Brand, Y.R. Fei, P. Holl, N. Thuerey, Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers, Adv. Neural Inf. Process. Syst. 33 (2020) 6111–6122.

[30] D. Kochkov, J.A. Smith, A. Alieva, Q. Wang, M.P. Brenner, S. Hoyer, Machine learning–accelerated computational fluid dynamics, Proc. Natl. Acad. Sci. USA 118 (21) (2021) e2101784118.

[31] M.B. Giles, M.C. Duta, J.-D. Muller, N.A. Pierce, Algorithm developments for discrete adjoint methods, AIAA J. 41 (2) (2003) 198–205.

[32] M.B. Giles, N.A. Pierce, An introduction to the adjoint approach to design, Flow Turbul. Combust. 65 (3) (2000) 393–415.

[33] A. Jameson, Aerodynamic design via control theory, J. Sci. Comput. 3 (1988) 233–260.

[34] E. Parish, K. Duraisamy, Quantification of turbulence modeling uncertainties using full field inversion, in: 22nd AIAA Computational Fluid Dynamics Conference, 2015, p. 2459.

[35] J.R. Holland, J.D. Baeder, K. Duraisamy, Towards integrated field inversion and machine learning with embedded neural networks for RANS modeling, in: AIAA Scitech 2019 Forum, American Institute of Aeronautics and Astronautics, eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2019-1884.

[36] O. Bidar, P. He, S. Anderson, N. Qin, An open-source adjoint-based field inversion tool for data-driven RANS modelling, https://doi.org/10.2514/6.2022-4125, 2022.

[37] O. Brenner, P. Piroozmand, P. Jenny, Efficient assimilation of sparse data into RANS-based turbulent flow simulations using a discrete adjoint method, J. Comput. Phys. 471 (2022) 111667.

[38] I.B.H. Saïdi, M. Schmelzer, P. Cinnella, F. Grasso, CFD-driven symbolic identification of algebraic Reynolds-stress models, J. Comput. Phys. 457 (2022) 111037.

[39] Y. Zhao, H.D. Akolekar, J. Weatheritt, V. Michelassi, R.D. Sandberg, RANS turbulence model development using CFD-driven machine learning, J. Comput. Phys. 411 (2020) 109413.

[40] C.A.M. Ströfer, H. Xiao, End-to-end differentiable learning of turbulence models from indirect observations, Theor. Appl. Mech. Lett. 11 (4) (2021) 100280.

[41] E.J. Parish, K. Duraisamy, Non-Markovian closure models for large eddy simulations using the Mori-Zwanzig formalism, Phys. Rev. Fluids 2 (1) (2017) 014604.

[42] J.A. Schaefer, A.W. Cary, M. Mani, P.R. Spalart, Uncertainty quantification and sensitivity analysis of SA turbulence model coefficients in two and three dimensions, in: 55th AIAA Aerospace Sciences Meeting, 2017, p. 1710.

[43] H. Xiao, P. Cinnella, Quantification of model uncertainty in RANS simulations: A review, Prog. Aerosp. Sci. 108 (2019) 1–31.

[44] J.-X. Wang, R. Sun, H. Xiao, Quantification of uncertainties in turbulence modeling: A comparison of physics-based and random matrix theoretic approaches, Int. J. Heat Fluid Flow 62 (2016) 577–592.

[45] M. Emory, J. Larsson, G. Iaccarino, Modeling of structural uncertainties in Reynolds-averaged Navier-stokes closures, Phys. Fluids 25 (11) (2013) 110822.

[46] C. Gorlé, G. Iaccarino, A framework for epistemic uncertainty quantification of turbulent scalar flux models for Reynolds-averaged Navier-stokes simulations, Phys. Fluids 25 (5) (2013) 055105.

[47] W.N. Edeling, G. Iaccarino, P. Cinnella, Data-free and data-driven RANS predictions with quantified uncertainty, Flow Turbul. Combust. 100 (3) (2018) 593–616.

[48] R.L. Thompson, A.A. Mishra, G. Iaccarino, W. Edeling, L. Sampaio, Eigenvector perturbation methodology for uncertainty quantification of turbulence models, Phys. Rev. Fluids 4 (4) (2019) 044603.

[49] J.-L. Wu, J.-X. Wang, H. Xiao, J. Ling, A priori assessment of prediction confidence for data-driven turbulence modeling, Flow Turbul. Combust. 99 (2017) 25–46.

[50] P.-S. Koutsourelakis, N. Zabaras, M. Girolami, Big data and predictive computational modeling, J. Comput. Phys. 321 (2016) 1252–1254.
[51] P.M. Gresho, D.K. Gartling, J. Torczynski, K. Cliffe, K. Winters, T. Garratt, A. Spence, J.W. Goodrich, Is the steady viscous incompressible two-dimensional flow over a backward-facing step at re = 800 stable?, Int. J. Numer. Methods Fluids 17 (6) (1993) 501–541.
[52] G. Alfonsi, Reynolds-averaged Navier-Stokes equations for turbulence modeling, Appl. Mech. Rev. 62 (Jul. 2009), https://doi.org/10.1115/1.3124648.
[53] S.E. Ahmed, S. Pawar, O. San, A. Rasheed, T. Iliescu, B.R. Noack, On closures for reduced order models – a spectrum of first-principle to machine-learned avenues, Phys. Fluids 33 (9) (2021) 091301, https://doi.org/10.1063/5.0061577, arXiv:2106.14954.
[54] O. San, R. Maulik, Neural network closures for nonlinear model order reduction, https://doi.org/10.48550/arXiv.1705.08532, May 2017.
[55] W. Snyder, C. Mou, H. Liu, O. San, R. De Vita, T. Iliescu, Reduced order model closures: a brief tutorial, arXiv:2202.14017, Feb. 2022.
[56] B.E. Launder, B.I. Sharma, Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc, Lett. Heat Mass Transf. 1 (2) (1974) 131–137, https://doi.org/10.1016/0094-4548(74)90150-7.
[57] D.C. Wilcox, Formulation of the k-ω turbulence model revisited, AIAA J. 46 (2008) 2823–2838, https://doi.org/10.2514/1.36541.
[58] D.C. Wilcox, et al., Turbulence Modeling for CFD, vol. 2, DCW Industries, La Canada, CA, 1998.
[59] C.G. Speziale, On nonlinear K-l and K-ϵ models of turbulence, J. Fluid Mech. 178 (1987) 459–475, https://doi.org/10.1017/S0022112087001319.
[60] T. Craft, B. Launder, K. Suga, Development and application of a cubic eddy-viscosity model of turbulence, Int. J. Heat Fluid Flow 17 (2) (1996) 108–115.
[61] B.E. Launder, G.J. Reece, W. Rodi, Progress in the development of a Reynolds-stress turbulence closure, J. Fluid Mech. 68 (3) (1975) 537–566.
[62] T.B. Gatski, C.G. Speziale, On explicit algebraic stress models for complex turbulent flows, J. Fluid Mech. 254 (1993) 59–78.
[63] S.S. Girimaji, Fully explicit and self-consistent algebraic Reynolds stress model, Theor. Comput. Fluid Dyn. 8 (6) (1996) 387–402.
[64] S.E. Ahmed, S. Pawar, O. San, A. Rasheed, T. Iliescu, B.R. Noack, On closures for reduced order models—a spectrum of first-principle to machine-learned avenues, Phys. Fluids 33 (9) (09 2021) 091301, https://doi.org/10.1063/5.0061577.
[65] J.-X. Wang, J.-L. Wu, H. Xiao, Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data, Phys. Rev. Fluids 2 (3) (2017) 034603.
[66] G. Iaccarino, A.A. Mishra, S. Ghili, Eigenspace perturbations for uncertainty estimation of single-point turbulence closures, Phys. Rev. Fluids 2 (2) (2017), https://doi.org/10.1103/PhysRevFluids.2.024605.
[67] A.A. Mishra, G. Iaccarino, Uncertainty estimation for Reynolds-averaged Navier-stokes predictions of high-speed aircraft nozzle jets, AIAA J. 55 (11) (2017) 3999–4004, https://doi.org/10.2514/1.J056059.
[68] C.M. Bishop, N.M. Nasrabadi, Pattern Recognition and Machine Learning, vol. 4, Springer, 2006.
[69] M.E. Tipping, C.M. Bishop, Probabilistic principal component analysis, J. R. Stat. Soc., Ser. B, Stat. Methodol. 61 (3) (1999) 611–622.
[70] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. USA 113 (15) (2016) 3932–3937.
[71] L. Felsberger, P. Koutsourelakis, Physics-constrained, data-driven discovery of coarse-grained dynamics, Commun. Comput. Phys. 25 (5) (2019) 1259–1301, https://doi.org/10.4208/cicp.OA-2018-0174.
[72] R.M. Neal, Bayesian Learning for Neural Networks, vol. 118, Springer Science & Business Media, 2012.
[73] A. Wu, M. Park, O. Koyejo, J.W. Pillow, Sparse Bayesian structure learning with dependent relevance determination prior, in: Proceedings of the 27th International Conference on Neural Information Processing Systems – Volume 1, NIPS'14, MIT Press, Cambridge, MA, USA, 2014, pp. 1628–1636.
[74] M.E. Tipping, The relevance vector machine, in: S.A. Solla, T.K. Leen, K.-R. Müller (Eds.), Advances in Neural Information Processing Systems, vol. 12, MIT Press, 2000, pp. 652–658.
[75] C. Bishop, Variational principal components, in: 1999 Ninth International Conference on Artificial Neural Networks ICANN 99, vol. 1, in: Conf. Publ., vol. 470, 1999, pp. 509–514.
[76] M.D. Hoffman, D.M. Blei, C. Wang, J. Paisley, Stochastic variational inference, J. Mach. Learn. Res. (2013).
[77] D.M. Blei, A. Kucukelbir, J.D. McAuliffe, Variational inference: A review for statisticians, J. Am. Stat. Assoc. 112 (518) (2017) 859–877.
[78] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.
[79] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, arXiv preprint, arXiv:1312.6114, 2013.
[80] J.M. Bardsley, Gaussian Markov random field priors for inverse problems, Inverse Probl. Imaging 7 (2) (2013) 397–416, https://doi.org/10.3934/ipi.2013.7.397.
[81] P.M. Nadge, R. Govardhan, High Reynolds number flow over a backward-facing step: structure of the mean separation bubble, Exp. Fluids 55 (1) (2014) 1–22.
[82] F. Pioch, J.H. Harmening, A.M. Müller, F.-J. Peitzmann, D. Schramm, O.e. Moctar, Turbulence modeling for physics-informed neural networks: comparison of different RANS models for the backward-facing step flow, Fluids 8 (2) (2023) 43, https://doi.org/10.3390/fluids8020043.
[83] H. Jasak, A. Jemcov, Z. Tukovic, et al., OpenFOAM: A C++ library for complex physics simulations, in: International Workshop on Coupled Methods in Numerical Dynamics, vol. 1000, 2007, pp. 1–20.
[84] F. Nicoud, F. Ducros, Subgrid-scale stress modelling based on the square of the velocity gradient tensor, Flow Turbul. Combust. 62 (3) (1999) 183–200.
[85] L. Engelmann, M. Ihme, I. Wlokas, A. Kempf, Towards the suitability of information entropy as an LES quality indicator, Flow Turbul. Combust. (2021) 1–33.
[86] I. Celik, Z. Cehreli, I. Yavuz, Index of resolution quality for large eddy simulations, 2005.
[87] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M.E. Rognes, G.N. Wells, The FEniCS project version 1.5, Arch. Numer. Softw. 3 (100) (2015).
[88] S.K. Mitusch, S.W. Funke, J.S. Dokken, dolfin-adjoint 2018.1: automated adjoints for FEniCS and Firedrake, J. Open Sour. Softw. 4 (38) (2019) 1292.
[89] E. Bingham, J.P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, N.D. Goodman, Pyro: Deep universal probabilistic programming, J. Mach. Learn. Res. 20 (1) (2019) 973–978.
[90] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019).
[91] B.F. Armaly, F. Durst, J. Pereira, B. Schönung, Experimental and theoretical investigation of backward-facing step flow, J. Fluid Mech. 127 (1983) 473–496.
[92] G. Biswas, M. Breuer, F. Durst, Backward-facing step flows for various expansion ratios at low and moderate Reynolds numbers, J. Fluids Eng. 126 (3) (2004) 362–374.
[93] J. Han, X.-H. Zhou, H. Xiao Vcnn-e, A vector-cloud neural network with equivariance for emulating Reynolds stress transport equations, arXiv preprint, arXiv: 2201.01287, 2022.
[94] P. Deuflhard, Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms, vol. 35, Springer Science & Business Media, 2005.
[95] J. Donea, A. Huerta, Finite Element Methods for Flow Problems, John Wiley & Sons, 2003.
[96] L.P. Franca, E.G.D. Do Carmo, The Galerkin gradient least-squares method, Comput. Methods Appl. Mech. Eng. 74 (1) (1989) 41–54.

B
Physics-Informed Tensor Basis Neural Network for Turbulence Closure Modeling

Physics-Informed Tensor Basis Neural Network for Turbulence Closure Modeling

Leon Riccius, Atul Agrawal & Phaedon-Stelios Koutsourelakis
Professorship of Data-driven Materials Modeling
School of Engineering and Design
Technical University of Munich
Boltzmannstr. 15, 85748 Garching, Germany
`{leon.riccius, atul.agrawal, p.s.koutsourelakis}@tum.de`

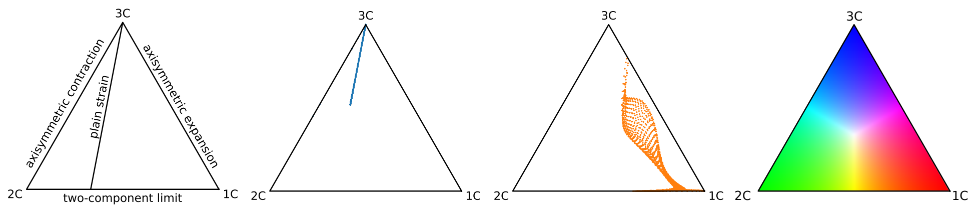
Abstract

Despite the increasing availability of high-performance computational resources, Reynolds-Averaged Navier-Stokes (RANS) simulations remain the workhorse for the analysis of turbulent flows in real-world applications. Linear eddy viscosity models (LEVM), the most commonly employed model type, cannot accurately predict complex states of turbulence. This work combines a deep-neural-network-based, nonlinear eddy viscosity model with turbulence realizability constraints as an inductive bias in order to yield improved predictions of the anisotropy tensor. Using visualizations based on the barycentric map, we show that the proposed machine learning method's anisotropy tensor predictions offer a significant improvement over all LEVMs in traditionally challenging cases with surface curvature and flow separation. However, this improved anisotropy tensor does not, in general, yield improved mean-velocity and pressure field predictions in comparison with the best-performing LEVM.

1 Introduction

The incompressible Navier-Stokes equations are vital for describing fluid motion at low Reynolds numbers, impacting fields like aircraft design and ocean current modeling. Turbulent flows are prohibitively expensive to resolve fully, and engineers often resort to reduced models such as RANS for efficiency. These models employ closures such as the Launder-Sharma $k-\epsilon$ [1] or Wilcox's $k-\omega$ [2], which rely on linear assumptions, limiting their accuracy in complex flow scenarios. Nonlinear models have been explored but face challenges. This contribution builds upon the resurgence of turbulence modeling research, instigated by data-driven approaches [3], in response to the stagnation seen in the 2000s after earlier advancements.

This work combines additional flow features derived by Wang and colleagues [4, 5] with the neural network architecture proposed by Ling et al. [6] to give point-based estimates of the anisotropy tensor appearing in the RANS closure. The training objectives are supplemented by a loss term penalizing predictions that violate the physical realizability constraints of turbulent states. The trained network proposed was tested on unseen flow scenarios and used as a source term in the RANS equations to produce estimates of the mean-flow quantities.



(a) Characteristic regions (b) RANS $k-\omega$ (c) DNS (d) RGB mapping

Figure 1: Barycentric map representing nature of turbulence of RANS $k-\omega$ **(a)** and DNS (b)

2 Physics-Informed Tensor Basis Neural Network

2.1 Reynolds stresses and realizability constraints

The Reynolds stress tensor $\tau_{ij} = \langle u_i u_j \rangle$, where $u_i = U_i - \langle U_i \rangle$ arises by time-averaging (mean indicated by $\langle \cdot \rangle$) of the Navier-Stokes equations and depends on the fluctuations u_i of the velocity field U_i. It can be decomposed into an isotropic δ_{ij} and anisotropic a_{ij} part which is given by $a_{ij} = \tau_{ij} - {}^2\!/_3 k \delta_{ij}$.The turbulent kinetic energy k is the trace of the Reynolds stresses.

The anisotropic a_{ij} has zero trace, and its normalized version b_{ij} is referred to as anisotropy tensor, i.e.:

$$b_{ij} = \frac{a_{ij}}{2k} = \frac{\tau_{ij}}{\langle u_k u_k \rangle} - \frac{1}{3}\delta_{ij}, \tag{1}$$

The Reynolds stress tensor is a symmetric, positive semi-definite second-order tensor with a non-negative determinant and trace. Following [7], the physical constraints, known also as realizability constraints, on the anisotropy tensor are:

$$-\frac{1}{3} \leq b_{\alpha\alpha} \leq \frac{2}{3} \quad \forall \alpha \in \{1,2,3\}, \qquad -\frac{1}{2} \leq b_{\alpha\beta} \leq \frac{1}{2} \quad \forall \alpha \neq \beta. \tag{2}$$

The barycentric map introduced in [8] uses an eigenvalue decomposition of $\boldsymbol{b}$ to define three fundamental states of turbulence. All other states of turbulence can be expressed as a linear combination of these three limiting states. The limiting states form a triangle of all admissible states; its vertices are defined by the realizability constraints (2). As demonstrated in Figure 1, each point in the barycentric triangle corresponds to a unique color. The mapping is given in the supplementary material 6.1.

2.2 Closure Model

While LEVMs assume $\boldsymbol{b}$ to be a linear function of the mean velocity gradient, a more general class of turbulence models can be formulated when dropping this assumption. The class of algebraic stress models is formed by nonlinear eddy viscosity models (NLEVM), which determine the Reynolds stresses from the local turbulent kinetic energy k, the eddy viscosity ϵ, and the mean velocity gradient. Pope [9] has shown that every second-order tensor that can be formed from the normalized mean rate of strain $\hat{\boldsymbol{S}} = {}^{\epsilon}\!/_{2k}(\nabla \langle \boldsymbol{U} \rangle + \nabla \langle \boldsymbol{U} \rangle^T)$ and the normalized rate of rotation $\hat{\boldsymbol{\Omega}} = {}^{\epsilon}\!/_{2k}(\nabla \langle \boldsymbol{U} \rangle - \nabla \langle \boldsymbol{U} \rangle^T)$ and fulfills these requirements is a linear combination of ten basis tensors $\mathcal{T}_{ij}^{(n)}$. The most general form of a NLEVM is given by

$$b_{ij} = \sum_{n=1}^{10} G^{(n)}(\lambda_1, ..., \lambda_5)\, \mathcal{T}_{ij}^{(n)}(\hat{S}_{ij}, \hat{\Omega}_{ij}), \tag{3}$$

where $G^{(n)}$ are the coefficients of the basis tensors and λ_k are the tensor invariants dependent on $\hat{\boldsymbol{S}}$ and $\hat{\boldsymbol{\Omega}}$. Ling et al. [6] introduced the Tensor Basis Neural Network (TBNN), which makes use of modern machine learning methods to learn these functions $G^{(n)}$ from high-fidelity fluid simulation data. Even though improved results compared to the $k-\epsilon$ model were reported, extracting enough information from these five invariants has proven difficult. This is especially true for flow cases with at least one direction of homogeneity, where invariants λ_3 and λ_4 vanish for the entire flow domain. Proof of this statement is given in the supplementary material 6.2. It is, however, possible to include more features from local flow quantities and derive more invariants while still employing the integrity basis formed by $\mathcal{T}^{(n)}$. This work, in part, follows the research of Wang et al. [10], who derived an extended feature set also considering the gradients of the turbulent kinetic energy and the pressure. The model was implemented in PyTorch [11] and can be accessed via github.com/pkmtum/PI_TBNN.

2.2.1 Enforcing Realizability Constraints in Training

Ling et al. [6] enforced the realizability constraints in post-processing by simply projecting the points onto the closest boundary of the barycentric triangle. We propose incorporating these constraints in the TBNN's training, which we then colloquially refer to as the Physics-Informed Tensor Basis Neural Network (PI-TBNN). The inequalities given in Eq. (2) can be transformed into contributions to the loss function via the penalty method. The additional term reflects inductive bias about the problem structure and essentially acts as a regularizer. In plain words, if training samples are outside the domain of realizable turbulence states, the penalty term will force them back in. The constraints are

$$\begin{aligned} c_1(\boldsymbol{b}) &= \min_{\alpha}(b_{\alpha\alpha}) - 1/3 < 0 \quad \forall\alpha \in \{1,2,3\}, & c_4(\boldsymbol{b}) &= 2|b_{12}| - (b_{11} + b_{22} + 2/3) < 0, \\ c_2(\boldsymbol{b}) &= (3|\phi_2| - \phi_2)/2 - \phi_1 < 0, & c_5(\boldsymbol{b}) &= 2|b_{13}| - (b_{11} + b_{33} + 2/3) < 0, \\ c_3(\boldsymbol{b}) &= 1/3 - \phi_2 < 0, & c_6(\boldsymbol{b}) &= 2|b_{23}| - (b_{22} + b_{33} + 2/3) < 0, \end{aligned} \tag{4}$$

where ϕ_i denotes the eigenvalues of $\boldsymbol{b}$ in order to distinguish them from the invariants. ϕ_1 and ϕ_2 are the largest and second-largest eigenvalues. The penalty term is then given by

$$\mathcal{L}(\hat{\boldsymbol{b}}(\boldsymbol{\lambda}, \boldsymbol{\theta})) = \beta \sum_{k=1}^{6} \max(0, c_k(\hat{\boldsymbol{b}}(\boldsymbol{\lambda}, \boldsymbol{\theta}))), \tag{5}$$

where $\boldsymbol{\lambda}$ is the collection of invariants, $\boldsymbol{\theta}$ are the NN parameters, and $\hat{\boldsymbol{b}}(\boldsymbol{\lambda}, \boldsymbol{\theta})$ is the predicted anisotropy tensor. The penalty coefficient β determines its impact on the loss function. The complete loss function, considering the MSE loss, the regularization, and penalty terms, is given by

$$E(\boldsymbol{\lambda}_i, \boldsymbol{\theta}) = \frac{1}{D} \sum_{i=1}^{D} \|\hat{\boldsymbol{b}}(\boldsymbol{\lambda}_i, \boldsymbol{\theta}) - \boldsymbol{b}_i\| + \frac{\alpha}{2} \|\boldsymbol{\theta}\|_2^2 + \frac{\beta}{D} \sum_{i=1}^{D} \sum_{k=1}^{6} \max(0, c_k(\hat{\boldsymbol{b}}(\boldsymbol{\lambda}_i, \boldsymbol{\theta}))), \tag{6}$$

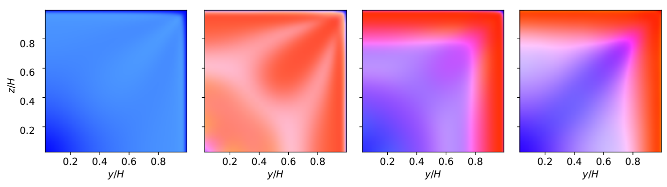
where $\boldsymbol{b}_i$ are the high-fidelity responses. The number of data points is denoted D. The coefficient α controls the degree of L_2 regularization.

3 Numerical Results

A total of four flow geometries were used as benchmarks in this work. The flow over periodic hills (PH) [12], the converging-diverging channel flow (CDC) [13], and the curved backward-facing step (CBFS) [14]. These exhibit adverse pressure gradients over curved surfaces, leading to flow separation and subsequent reattachment. The data set was also extended to include the square duct (SD) flow case [15]. This scenario clearly illustrates the limitations of LEVMs and is well-suited for investigating the forward propagation of the predicted anisotropy tensor. All flow cases were replicated in OpenFOAM [16] to obtain the baseline RANS data. The flow case setup is analogous to [17]. A detailed description of the data sets is given in the supplementary material 6.3.

3.1 Anisotropy Tensor Prediction

The PI-TBNN was tested on flow cases it had not seen during training. They differ either in geometry or Reynolds number from the training data. Figure 2 compares the anisotropy tensors for square duct (SD) using the barycentric colormap. Only the TBNN with the extended feature set can accurately reproduce the state of turbulence for this flow case. A similar picture arises on the PH geometry in Figure 3, where the $k - \omega$ model cannot capture 1C turbulence and axisymmetric expansion at the top and the bulk of the flow domain, respectively. The PI-TBNN, however, does exhibit such characteristics. For all test cases considered, the PI-TBNN achieves about 70% reduction of the RMSE compared to the baseline $k - \omega$ model and 50% reduction of the RMSE compared to [6] (see Table 1).



(a) LEVM $k-\omega$ (b) PI-TBNN1 (c) PI-TBNN2 (d) DNS

Figure 2: Stress types of $k-\omega$ **(a)**, PI-TBNN with FS1 **(b)**, FS[1-3] **(c)**, and DNS **(d)** for SD.

Table 1: RMSE of $\boldsymbol{b}$ from RANS, PI-TBNN, and TBNN predictions for the three test cases.

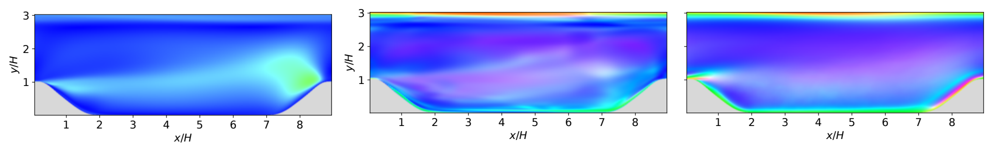
Flow case	LEVM $k-\omega$	PI-TBNN, FS1	PI-TBNN, FS[1-3]	TBNN Ling [6]
Square duct	0.2175	0.0992	0.0663	0.14
Periodic hills	0.1016	0.0628	0.0419	
Curved backward-facing step	0.1173	0.0619	0.0414	

3.2 Anisotropy Tensor Propagation

While some applications involving wall shear stress computations may directly benefit from an improved prediction of $\boldsymbol{b}$, the quantities of interest are usually the mean velocity and pressure fields. Hence, with the PI-TBNN model, the Reynolds equations were solved for the mean velocity and pressure fields. Table 2 shows that the PI-TBNN outperforms the $k-\epsilon$ model for the PH and CBFS geometries by a small margin and yields better in-plane prediction for the square duct flow case. The most accurate model, however, remains the $k-\omega$ model for all three test geometries. Surprisingly, on the CBFS geometry, the PI-TBNN shows the largest discrepancies in the region of the flow separation, as can be seen in Figure 4. The $k-\omega$ model even beats the mean field resulting from using the anisotropy tensor from the DNS on the periodic hills test case, indicating that an improved anisotropy tensor does not necessarily lead to improved mean velocity and pressure fields (behavior also reported in [18, 19]). Both [6] and [20] reported improvements in the mean-field prediction over a LEVM but used the $k-\epsilon$ as the baseline LEVM, which shows a larger discrepancy from the ground truth than the $k-\omega$ model, i.e. the best performing model of its class.

4 Conclusion

We introduced the PI-TBNN, which extended the TBNN framework with an extensive feature set and an inductive bias in the form of a physics-informed addition to the loss function. The addition of features was motivated both analytically—showing that the number of distinct invariants for 2D flow scenarios is three, not five—and empirically through improved predictions. It has been shown that the new approach yields more accurate predictions of the anisotropy tensor than the original TBNN of Ling et al. [6]. The improvements were illustrated with the barycentric colormap and quantified by comparing RMSEs. It is, however, limited in its predictive capabilities of the mean velocity and pressure fields. While it still outperformed the widely popular $k-\epsilon$ model on geometries with flow separation, it consistently fails to compete with the $k-\omega$ model. The rather large discrepancy of the RANS using the DNS anisotropy tensor indicates that it is more beneficial to train models that not only aim at improving predictions of the Reynolds stresses but instead target the mean field quantities directly, e.g., as in [21, 22, 23].

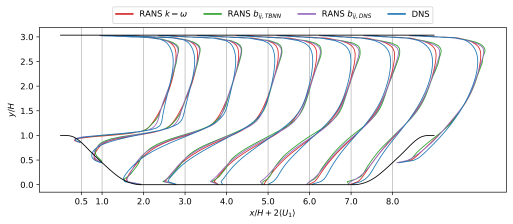


(a) LEVM $k-\omega$ (b) PI-TBNN (c) DNS

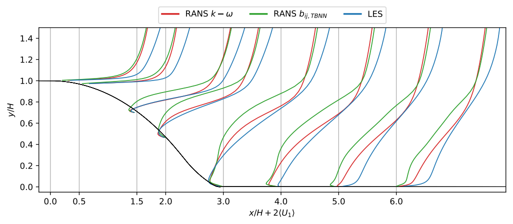
Figure 3: Visualization of stress types of LEVM $k-\omega$ **(a)**, PI-TBNN **(b)**, and DNS **(c)**

Table 2: RMSE of $\boldsymbol{U}$ for RANS with anisotropy tensor from $k-\omega$, $k-\epsilon$, PI-TBNN, and DNS. Reference velocity fields come from DNS.

Flow case	$k-\omega$	$k-\epsilon$	$\boldsymbol{b}_{\mathrm{PI-TBNN}}$	$\boldsymbol{b}_{\mathrm{DNS}}$
Square duct RMSE($\boldsymbol{U}$)	0.0496	0.0667	0.0716	0.0322
Square duct RMSE($[U_2, U_3]$)	0.0066	0.0066	0.0045	0.0025
Periodic hills RMSE($\boldsymbol{U}$)	0.0375	0.0545	0.0541	0.0465
Curved backward-facing step RMSE($\boldsymbol{U}$)	0.0609	0.0883	0.0868	



(a) Periodic hills



(b) Curved backward-facing step

Figure 4: Streamwise mean velocity profiles at specific x-locations of PH **(a)**, and CBFS **(b)**.

5 Broader Impact

Turbulence is a key physical characteristic of a broad range of fluid flows. Understanding this phenomenon is crucial for complex designs, environmental modeling, and many more engineering applications. Computational power has increased massively in the past decades, enabling scale-resolving simulations like direct numerical simulations of a number of canonical turbulent flows. However, fast approximations like the RANS continue to remain essential for industrial applications, whose accuracy hinges heavily on turbulence closure models.

The presented research serves as an extension to the state-of-the-art data-driven turbulence closure model proposed by [6]. By combining a deep neural network with an inductive bias informed by turbulence realizability constraints, plus an extensive feature set, the PI-TBNN showed considerable improvements. These improvements showcased through barycentric colormap visualizations and quantified reductions in RMSE signify a step forward in our ability to capture complex turbulence states, particularly in scenarios involving surface curvature and flow separation.

However, the study also sheds light on the nuanced relationship between improved anisotropy tensor predictions and the ultimate goal of predicting mean velocity and pressure fields. While the PI-TBNN excels in enhancing anisotropy tensor predictions, it does not consistently outperform the established $k-\omega$ model in mean-field predictions, underscoring the need for further exploration in this area.

We do not see any direct ethical concerns associated with this research. The impact on society is primarily through the over-arching context of research using machine learning to improve our general understanding of turbulence in fluids.

References

[1] B. E. Launder and B. I. Sharma. “Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc”. In: *Letters in Heat and Mass Transfer* 1.2 (Nov. 1974), pp. 131–137. ISSN: 00944548. DOI: 10.1016/0094-4548(74)90150-7.

[2] David C. Wilcox. “Formulation of the k-ω turbulence model revisited”. In: *AIAA Journal*. Vol. 46. 11. Nov. 2008, pp. 2823–2838. DOI: 10.2514/1.36541. URL: https://arc.aiaa.org/doi/abs/10.2514/1.36541.

[3] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. “Turbulence Modeling in the Age of Data”. In: *Annual Review of Fluid Mechanics* 51.1 (2019), pp. 357–377. ISSN: 0066-4189. DOI: 10.1146/annurev-fluid-010518-040547.

[4] Jian Xun Wang and Heng Xiao. “Data-driven CFD modeling of turbulent flows through complex structures”. In: *International Journal of Heat and Fluid Flow* 62 (2016), pp. 138–149. ISSN: 0142727X. DOI: 10.1016/j.ijheatfluidflow.2016.11.007.

[5] Jian Xun Wang, Jin Long Wu, and Heng Xiao. “Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data”. In: *Physical Review Fluids* 2.3 (2017), p. 34603. ISSN: 2469990X. DOI: 10.1103/PhysRevFluids.2.034603.

[6] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance”. In: *Journal of Fluid Mechanics* 807 (2016), pp. 155–166. ISSN: 14697645. DOI: 10.1017/jfm.2016.615.

[7] U. Schumann. “Realizability of Reynolds-stress turbulence models”. In: *Physics of Fluids* 20.5 (1977), pp. 721–725. ISSN: 10706631. DOI: 10.1063/1.861942. URL: https://doi.org/10.1063/1.861942.

[8] S. Banerjee et al. “Presentation of anisotropy properties of turbulence, invariants versus eigenvalue approaches”. In: *Journal of Turbulence* 8 (2007), pp. 1–27. ISSN: 14685248. DOI: 10.1080/14685240701506896. URL: https://www-tandfonline-com.eaccess.ub.tum.de/doi/abs/10.1080/14685240701506896.

[9] S. B. Pope. “A more general effective-viscosity hypothesis”. In: *Journal of Fluid Mechanics* 72.2 (Nov. 1975), pp. 331–340. ISSN: 14697645. DOI: 10.1017/S0022112075003382. URL: http://www.journals.cambridge.org/abstract_S0022112075003382.

[10] Jian Xun Wang et al. *A comprehensive physics-informed machine learning framework for predictive turbulence modeling*. 2017. DOI: 10.1103/PhysRevFluids.3.074602.

[11] Adam Paszke et al. *PyTorch: An imperative style, high-performance deep learning library*. Dec. 2019. URL: http://arxiv.org/abs/1912.01703.

[12] C P Mellen, J Frohlic, and W. Rodi. “Large Eddy Simulation of the flow over periodic hills”. In: *16th IMACS World Congress* (2000), pp. 3–8.

[13] Jean Philippe Laval and Matthieu Marquillie. “Direct numerical simulations of converging–diverging channel flow”. In: *ERCOFTAC Series*. Vol. 14. Springer Netherland, 2011, pp. 203–209. ISBN: 9789048196029. DOI: 10.1007/978-90-481-9603-6{_}21.

[14] Yacine Bentaleb, Sylvain Lardeau, and Michael A. Leschziner. “Large-eddy simulation of turbulent boundary layer separation from a rounded step”. In: *Journal of Turbulence* 13 (Jan. 2012), pp. 1–28. ISSN: 14685248. DOI: 10.1080/14685248.2011.637923. URL: https://www.tandfonline.com/doi/full/10.1080/14685248.2011.637923.

[15] Alfredo Pinelli et al. “Reynolds number dependence of mean flow structure in square duct turbulence”. In: *Journal of Fluid Mechanics* 644 (2010), pp. 107–122. ISSN: 00221120. DOI: 10.1017/S0022112009992242. URL: http://openaccess.city.ac.uk/.

[16] H. G. Weller et al. “A tensorial approach to computational continuum mechanics using object-oriented techniques”. In: *Computers in Physics* 12.6 (Dec. 1998), p. 620. ISSN: 08941866. DOI: 10.1063/1.168744. URL: http://scitation.aip.org/content/aip/journal/cip/12/6/10.1063/1.168744.

[17] Leon Riccius. *Machine Learning Augmented Turbulence Modelling for the Reynolds Stress Closure Problem*. Munich, Mar. 2021.

[18] Salar Taghizadeh, Freddie D. Witherden, and Sharath S. Girimaji. "Turbulence closure modeling with data-driven techniques: physical compatibility and consistency considerations". In: *New Journal of Physics* 22.9 (Sept. 2020), p. 093023. ISSN: 1367-2630. DOI: 10.1088/1367-2630/ABADB3. URL: https://iopscience.iop.org/article/10.1088/1367-2630/abadb3%20https://iopscience.iop.org/article/10.1088/1367-2630/abadb3/meta.

[19] Karthik Duraisamy. "Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence". In: *Physical Review Fluids* 6.5 (May 2021), p. 050504. ISSN: 2469990X. DOI: 10.1103/PHYSREVFLUIDS.6.050504/FIGURES/5/MEDIUM. URL: https://journals.aps.org/prfluids/abstract/10.1103/PhysRevFluids.6.050504.

[20] Jin Long Wu, Heng Xiao, and Eric Paterson. "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework". In: *Physical Review Fluids* 7.3 (2018), p. 74602. ISSN: 2469990X. DOI: 10.1103/PhysRevFluids.3.074602.

[21] Atul Agrawal and Phaedon-Stelios Koutsourelakis. "A probabilistic, data-driven closure model for RANS simulations with aleatoric, model uncertainty". In: *arXiv preprint* (July 2023). DOI: https://doi.org/10.48550/arXiv.2307.02432. URL: http://arxiv.org/abs/2307.02432.

[22] Michael E. Hayek, Qiqi Wang, and Gregory M. Laskowski. "Adjoint-based optimization of RANS eddy viscosity model for U-bend channel flow". In: *AIAA Aerospace Sciences Meeting, 2018*. 210059. Reston, Virginia: American Institute of Aeronautics and Astronautics Inc, AIAA, Jan. 2018. ISBN: 9781624105241. DOI: 10.2514/6.2018-2091. URL: https://arc.aiaa.org/doi/10.2514/6.2018-2091.

[23] Eric J Parish and Karthik Duraisamy. "A paradigm for data-driven predictive modeling using field inversion and machine learning". In: *Journal of Computational Physics* 305 (2016), pp. 758–774. ISSN: 10902716. DOI: 10.1016/j.jcp.2015.11.012. URL: http://www.elsevier.com/open-access/userlicense/1.0/.

6 Supplementary Material

6.1 RGB colormap

As demonstrated in Figure 1, each point in the barycentric triangle corresponds to a unique color. The mapping from the barycentric coordinates to the RGB values follows

$$\begin{bmatrix}\mathrm{R}\\\mathrm{G}\\\mathrm{B}\end{bmatrix} = \frac{1}{\max C_{ic}}\left(C_{1c}\begin{bmatrix}1\\0\\0\end{bmatrix} + C_{2c}\begin{bmatrix}0\\1\\0\end{bmatrix} + C_{3c}\begin{bmatrix}0\\0\\1\end{bmatrix}\right) \quad \text{for } i \in \{1,2,3\}. \tag{7}$$

6.2 Scalar invariants

Two of the five scalar invariants (λ_3, λ_4) are zero for flows with one direction of homogeneity. The two invariants read

$$\lambda_3 = \mathrm{tr}(\hat{\boldsymbol{S}}^3), \qquad \lambda_4 = \mathrm{tr}(\hat{\boldsymbol{\Omega}}^2\hat{\boldsymbol{S}}). \tag{8}$$

Assuming the flow is homogeneous in z-direction, the partial derivatives of the mean velocity with respect to z vanish, and the mean rate of strain and rotation read

$$\hat{S}_{ij} = \frac{1}{2}\frac{k}{\epsilon}\begin{bmatrix} 2\frac{\partial\langle U_x\rangle}{\partial x} & \frac{\partial\langle U_x\rangle}{\partial y} + \frac{\partial\langle U_y\rangle}{\partial x} \\ \frac{\partial\langle U_y\rangle}{\partial x} + \frac{\partial\langle U_x\rangle}{\partial y} & 2\frac{\partial\langle U_y\rangle}{\partial y}\end{bmatrix}, \quad \hat{\Omega}_{ij} = \frac{1}{2}\frac{k}{\epsilon}\begin{bmatrix} 0 & \frac{\partial\langle U_x\rangle}{\partial y} - \frac{\partial\langle U_y\rangle}{\partial x} \\ \frac{\partial\langle U_y\rangle}{\partial x} - \frac{\partial\langle U_x\rangle}{\partial y} & 0\end{bmatrix}. \tag{9}$$

The incompressibility constraint of the Reynolds equations reduces to

$$\frac{\partial\langle U_i\rangle}{\partial x_i} = \mathrm{tr}\left(\frac{\partial\langle U_i\rangle}{\partial x_j}\right) = \frac{\partial\langle U_x\rangle}{\partial x} + \frac{\partial\langle U_y\rangle}{\partial y} = 0. \tag{10}$$

When using the simplified expressions of $\hat{\boldsymbol{S}}$ and $\hat{\boldsymbol{\Omega}}$ in combination with the incompressibility constraint, invariant λ_3 is given by

$$\operatorname{tr}(\hat{S}^3_{ij}) = \tag{11}$$

$$\frac{k^3}{8\epsilon^3}\operatorname{tr}\left(\begin{bmatrix} \hat{S}_{11}(\hat{S}^2_{11}+\hat{S}^2_{12})+\hat{S}_{12}(\hat{S}_{11}\hat{S}_{12}+\hat{S}_{12}\hat{S}_{22}) & \hat{S}_{12}(\hat{S}^2_{11}+\hat{S}^2_{12})+\hat{S}_{22}(\hat{S}_{11}\hat{S}_{12}+\hat{S}_{12}\hat{S}_{22}) \\ \hat{S}_{11}(\hat{S}_{11}\hat{S}_{12}+\hat{S}_{12}\hat{S}_{22})+\hat{S}_{12}(\hat{S}^2_{12}+\hat{S}^2_{22}) & \hat{S}_{22}(\hat{S}^2_{12}+\hat{S}^2_{22})+\hat{S}_{12}(\hat{S}_{11}\hat{S}_{12}+\hat{S}_{12}\hat{S}_{22}) \end{bmatrix}\right)$$

$$= \frac{k^3}{8\epsilon^3}\left(\hat{S}_{11}(\hat{S}^2_{11}+\hat{S}^2_{12})+2\hat{S}^2_{12}\underbrace{(\hat{S}_{11}+\hat{S}_{22})}_{=0}+\hat{S}_{22}(\hat{S}^2_{22}+\hat{S}^2_{12})\right), \tag{12}$$

$$= \frac{k^3}{8\epsilon^3}\left(\hat{S}_{11}(\hat{S}^2_{11}+\hat{S}^2_{12})+\hat{S}_{22}(\hat{S}^2_{11}+\hat{S}^2_{12})\right), \tag{13}$$

$$= \frac{k^3}{8\epsilon^3}\left(\underbrace{(\hat{S}_{11}+\hat{S}_{22})}_{=0}(\hat{S}^2_{11}+\hat{S}^2_{12})\right) = 0. \tag{14}$$

The derivation of invariant λ_4 is more straightforward and thus written in terms of the mean velocity gradient. It is given by

$$\operatorname{tr}(\hat{\Omega}^2_{ij}\hat{S}_{jk}) = \operatorname{tr}\left(\frac{k^3}{8\epsilon^3}\left(\frac{\partial\langle U_x\rangle}{\partial y}-\frac{\partial\langle U_y\rangle}{\partial x}\right)^2\begin{bmatrix} 2\frac{\partial\langle U_x\rangle}{\partial x} & \frac{\partial\langle U_x\rangle}{\partial y}+\frac{\partial\langle U_y\rangle}{\partial x} & 0 \\ \frac{\partial\langle U_y\rangle}{\partial x}+\frac{\partial\langle U_x\rangle}{\partial y} & 2\frac{\partial\langle U_y\rangle}{\partial y} & 0 \\ 0 & 0 & 0 \end{bmatrix}\right) \tag{15}$$

$$= \frac{k^3}{4\epsilon^3}\left(\frac{\partial\langle U_x\rangle}{\partial y}-\frac{\partial\langle U_y\rangle}{\partial x}\right)\underbrace{\left(\frac{\partial\langle U_x\rangle}{\partial x}+\frac{\partial\langle U_y\rangle}{\partial y}\right)}_{=0} = 0. \tag{16}$$

6.3 Data set

The high-fidelity direct numerical simulation (DNS)/large eddy simulation (LES) data are used for the training. Out of the two available DNS for the CDC, the one at $\mathrm{Re} = 7900$ was used for estimating the regularization parameters. The higher Reynolds number simulation at $\mathrm{Re} = 12600$ was used for training and validation, along with the DNS at $\mathrm{Re} = 2800$ and LES at $\mathrm{Re} = 10595$ for PH and the DNS for SD at $\mathrm{Re} = \{2000, 2400, 2900, 3200\}$. The data set was split into training and validation sets at a ratio of $70/30$. Therefore, the total number of data points available for training was 26600.

The testing set consists of three flow geometries (SD, PH, and CBFS). Two of these geometries, SD and PH, are also part of the training and validation set, however, at different Reynolds numbers. The periodic hills case at $\mathrm{Re} = 5600$ was selected to investigate the interpolation properties — the ML model has previously seen this flow geometry and is expected to yield good predictions. The square duct is a canonical flow case that clearly illustrates the deficiencies of the LEVMs and is suitable to present difficulties of propagating the Reynolds stresses to the flow field. Finally, the curved backward-facing step tests the ML model's extrapolation capabilities.

C

From concrete mixture to structural design - a holistic optimization procedure in the presence of uncertainties

Data-Centric Engineering (2024), 1–32
doi:10.1017/dce.2024.18

RESEARCH ARTICLE

From concrete mixture to structural design—a holistic optimization procedure in the presence of uncertainties

Atul Agrawal[1], Erik Tamsen[2], Jörg F. Unger[2] and Phaedon-Stelios Koutsourelakis[1,3]

[1]Data-Driven Materials Modeling, Technische Universität München, Garching, Germany
[2]Modeling and Simulation, Bundesanstalt für Materialforschung und -prüfung, Berlin, Germany
[3]Munich Data Science Institute, Garching bei München, Germany
Corresponding author: Jorg Unger; Email: joerg.unger@bam.de

Received: 28 December 2023; **Revised:** 07 May 2024; **Accepted:** 27 May 2024

Keywords: black-box optimization under uncertainty; mix design; performance oriented design; precast concrete; probabilistic machine learning; sustainable material design

Abstract

We propose a systematic design approach for the precast concrete industry to promote sustainable construction practices. By employing a holistic optimization procedure, we combine the concrete mixture design and structural simulations in a joint, forward workflow that we ultimately seek to invert. In this manner, new mixtures beyond standard ranges can be considered. Any design effort should account for the presence of uncertainties which can be aleatoric or epistemic as when data are used to calibrate physical models or identify models that fill missing links in the workflow. Inverting the causal relations established poses several challenges especially when these involve physics-based models which more often than not, do not provide derivatives/sensitivities or when design constraints are present. To this end, we advocate Variational Optimization, with proposed extensions and appropriately chosen heuristics to overcome the aforementioned challenges. The proposed approach to treat the design process as a workflow, learn the missing links from data/models, and finally perform global optimization using the workflow is transferable to several other materials, structural, and mechanical problems. In the present work, the efficacy of the method is exemplarily illustrated using the design of a precast concrete beam with the objective to minimize the global warming potential while satisfying a number of constraints associated with its load-bearing capacity after 28 days according to the Eurocode, the demolding time as computed by a complex nonlinear finite element model, and the maximum temperature during the hydration.

Impact Statement

This article provides a framework to reduce the global warming potential of civil structures such as bridges, dams, or buildings while satisfying constraints relating to structural efficiency. The framework combines mixture design and structural simulation in a joint workflow to enable optimization. Advanced optimization algorithm with appropriate extensions and heuristics are employed to account for stochasticity of workflow, nonlinear constraints, and lack of derivatives of the workflow. Some relations in the workflow are not known a priori in the literature. These are learned by employing advanced probabilistic machine learning algorithms trained using the fusion of noisy data and physical models. Various forms of uncertainty arising due to noise in the data or

This research article was awarded Open Data and Open Materials badges for transparent practices. See the Data Availability Statement for details.

incompleteness of data are systematically incorporated into the framework. The big idea here is to create structures with a smaller environmental footprint without compromising their strength while being cost-efficient for the manufacturer. The proposed holistic approach is demonstrated on a specific design problem, but should serve as a template that can be readily adapted to other design problems.

1. Introduction

Precast concrete elements play a critical role in achieving efficient, low-cost, and sustainable structures. The controlled manufacturing environment allows for higher quality products and enables the mass production of such elements. In the standard design approach, engineers or architects select the geometry of a structure, estimate the loads, choose mechanical properties, and design the element accordingly. If the results are unsatisfactory, the required mechanical properties are iteratively adjusted, aiming to improve the design. This approach is adequate when the choice of mixtures is limited, and the expected concrete properties are well known. There are various published methods to automate this process and optimize the beam design at this level. Computer-aided beam design optimization dates back at least 50 years (e.g., Haung and Kirmser, 1967).

Generally, the objective is to reduce costs, with the design variables being the beam geometry, the amount and location of the reinforcement, and the compressive strength of the concrete (Chakrabarty, 1992; Coello et al., 1997; Pierott et al., 2021; Shobeiri et al., 2023). Most publications focus on analytical functions based on well-known, empirical rules of thumb. In recent years, the use of alternative binders in the concrete mix design has increased, mainly to reduce the environmental impact and cost of concrete but also to improve and modify specific properties. This is a challenge as the concrete mix is no longer a constant and is itself subject to an optimization. Known heuristics might no longer apply to the new materials, and old design approaches might fail to produce optimal results. In addition, it is not desirable to choose from a predetermined set of possible mixes, as this would lead to either an overwhelming number of required experiments or a limiting subset of the possible design space.

In the existing literature on the optimization of the concrete mix design (Lisienkova et al., 2021; Kondapally et al., 2022; Forsdyke et al., 2023), the objective is either to improve mechanical properties like durability within constraints or to minimize, for example, the amount of concrete while keeping other properties above a threshold. A first step to address these limitations is incorporating the compressive strength during optimization in the beam design phase. Higher compressive strength usually correlates with a larger amount of cement and, therefore, higher cost as well as a higher global warming potential (GWP). This approach has shown promising results in achieving improved structural efficiency while considering environmental impact (dos Santos et al., 2023). To be able to find a part specific optimum, individual data of the manufacturer and specific mix options must be integrated. Therefore, there is still a need for a comprehensive optimization procedure that can seamlessly integrate concrete mix design and structural simulations, ensuring structurally sound and buildable elements with minimized environmental impact for part specific data.

When designing elements subjected to various requirements, both on the material and structural levels, including workability of the fresh concrete, durability of the structure, maximum acceptable temperature, minimal cost, and GWP, the optimal solution is not apparent and will change depending on each individual project. In this article, we present a holistic optimization procedure that combines physics-based models and experimental data in order to enable the optimization of the concrete mix design in the presence of uncertainty, with an objective to minimize the GWP. In particular, we employ structural simulations as constraints to ensure structural integrity, limit the maximum temperature, and ensure an adequate time of demolding.

By integrating the concrete mixture optimization and structural design processes, engineers can tailor the concrete properties to meet the specific requirements of the customer and manufacturer. This approach opens up possibilities for performance prediction and optimization for new mixtures that fall outside the standard range of existing ones. To the best of our knowledge, there are no published works that combine the material and structural levels in one flexible optimization framework. In addition to changing the order of the design steps, the proposed framework allows to directly integrate experimental data and propagate the identified uncertainties. This allows a straightforward integration of new data and quantification of uncertainties regarding the predictions. The proposed framework consists of three main parts. The first part introduces an automated and reproducible probabilistic machine learning-based parameter identification method to calibrate the models by using experimental data. The second part focuses on a black-box optimization method for non-differentiable functions, including constraints. The third part presents a flexible workflow combining the models and functions required for the respective problem.

To carry out black-box optimization, we advocate the use of Variational Optimization (Staines and Barber, 2013; Bird et al., 2018), which uses stochastic gradient estimators for black-box functions. We utilize this with appropriate enhancements in order to account for the stochastic, nonlinear constraints. Our choice is motivated by three challenges present in the workflow describing the physical process. First, we are limited by the availability of only black-box evaluations of the physical workflow. In many real-world cases involving physics-based solvers/simulators in the optimization process, one resorts to gradient-free optimization (Moré and Wild, 2009; Snoek et al., 2012). However, the gradient-free methods perform poorly on high-dimensional parametric spaces (Moré and Wild, 2009). Also, it requires more functional evaluations to reach the optimum as compared to gradient-based methods. Recently, stochastic gradient estimators (Mohamed et al., 2020) have been used to estimate gradients of black-box functions and, hence, perform gradient-based optimization (Ruiz et al., 2018; Louppe et al., 2019; Shirobokov et al., 2020). However, they do not account for the constraints. Second, the presence of nonlinear constraints makes the optimization challenging. Popular gradient-free methods like constrained Bayesian optimization (cBO) (Gardner et al., 2014) and COBYLA (Powell, 1994) pose significant challenges when (non)linear constraints are involved (Audet and Kokkolaras, 2016; Menhorn et al., 2017; Agrawal et al., 2023). The third challenge is the stochasticity of the workflow, discussed in the following paragraph.

The physical workflow comprising physics-based models to link design variables with the objective and constraints poses an information flow-related challenge. Some links leading to the objective/constraints are not known a priori in the literature, thus hindering the optimization process. We propose a method to learn these missing links, parameterized by an appropriate neural network, with the help of (noisy) experimental data and physical models. The unavoidable noise in the data introduces aleatoric uncertainty, or its incompleteness introduces epistemic uncertainty. To account for the presence of these uncertainties, we advocate the links to be *probabilistic.* The learned probabilistic links tackle the information bottleneck; however, it introduces random parameters in the physical workflow, thus necessitating optimization under uncertainty (OUU) (Acar et al., 2021; Martins and Ning, 2021; Qiu et al., 2021). Deterministic inputs can lead to a poor-performing design, which OUU tries to tackle by producing a robust and reliable design that is less sensitive to inherent variability. This paradigm of fusing data and physical models to train machine-learning models has been extensively used across engineering and physics in recent years (Koutsourelakis et al., 2016; Fleming, 2018; Baker et al., 2019; Karniadakis et al., 2021; Karpatne et al., 2022; Lucor et al., 2022; Agrawal and Koutsourelakis, 2023; Forsdyke et al., 2023), colloquially referred to as scientific machine learning (SciML). In contrast to traditional machine learning areas where big data are generally available, engineering and physical applications generally suffer from a lack of data, further complicated by experimental noise. SciML has shown promise in addressing this lack of data. We summarize our key contributions below:

- We present an inverse design approach for the precast concrete industry to promote sustainable construction practices.
- To achieve the desired goals, we propose an algorithmic framework with two main components. (a) an optimization algorithm that accounts for the presence of various uncertainties, nonlinear constraints, and lack of derivatives and (b) a probabilistic machine learning algorithm that learns the missing relations to enable the optimization, by combining noisy, experimental data with physical

models. The algorithmic framework is transferable to several other material, structural, and mechanical design problems.
- To assist the optimization procedure, we propose an automated workflow that combines concrete mixture design and structural simulation.
- We demonstrate the effectiveness of the algorithmic framework and the workflow, on a precast concrete beam element. We learn the missing probabilistic link between the mixture design variables and finite element (FE) model parameters describing the concrete hydration and homogenization procedure. Subsequently, we optimize mixture design and beam geometry in the presence of uncertainties, with the goal of reducing the GWP while employing structural simulations as constraints to ensure safe and reliable design.

The structure of the rest of this article is as follows. Section 2.1 describes the proposed design approach, and Section 2.2 describes the physical material models and the applied assumptions. Section 2.3 presents the details of the experimental data. Section 2.4 provides an overview of the aforementioned probabilistic links and the optimization procedure. Section 2.5 talks about the methodology employed to learn the probabilistic links based on the experimental data and the physical models. Then Section 2.6 describe the details of the proposed black-box optimization algorithm. In Section 3, we showcase and discuss the results of the numerical experiments combining all the parts, the experimental data, the physical models, the identification of the probabilistic links, and the optimization framework. Finally, in Section 4, we summarize our findings and discuss possible extensions.

1.1. Demonstration problem

In this work, a well-known example of a simply supported, reinforced, rectangular beam has been chosen. The design problem was originally published in Everard and Tanner (1966) and illustrated in Figure 1.

It has been used to showcase different optimization schemes (e.g., Chakrabarty, 1992; Coello et al., 1997; Pierott et al., 2021). As opposed to the literature where the optimization is often related to costs, we aim to reduce the overall GWP of the part. This objective is particularly meaningful as the cement industry accounts for approximately 8% of the total anthropogenic GWP (Miller et al., 2016). Reducing the environmental impact of concrete production becomes crucial in the pursuit of sustainable construction practices. In addition, the reduction of the amount of cement in concrete is also correlated to the reduction of cost, as cement is generally the most expensive component of the concrete mix (Paya-Zaforteza et al., 2009). There are three direct ways to reduce the GWP of a given concrete part. First, replace the cement with a substitute with a lower carbon footprint. This usually changes mechanical properties and, in particular, their temporal evolution. Second, increase the amount of aggregates, therefore reducing the cement per volume. This also changes effective properties and needs to be balanced with the workability and the limits due to the applications. Third, decrease the overall volume of concrete by improving the

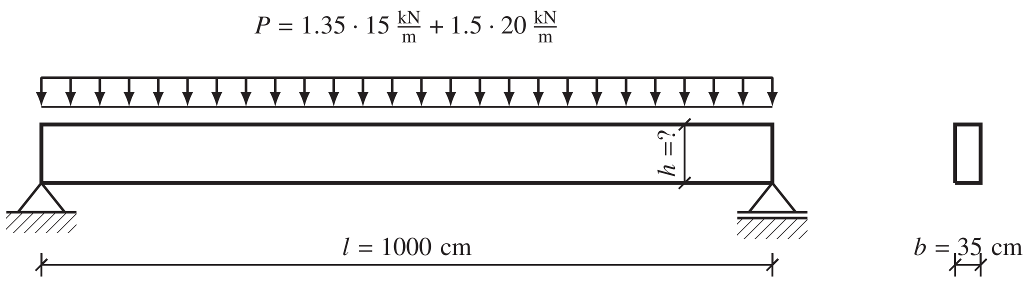

$P = 1.35 \cdot 15\,\frac{\mathrm{kN}}{\mathrm{m}} + 1.5 \cdot 20\,\frac{\mathrm{kN}}{\mathrm{m}}$
$h = ?$
$l = 1000$ cm
$b = 35$ cm

Figure 1. *Geometry of the design problem of a bending beam with a constant distributed load (dead load and live load with safety factors of 1.35 and 1.5) and a rectangular cross section. The design variable, beam height is denoted by h.*

topology of the part. In addition, when analyzing the whole life cycle of a structure, both cost and GWP can be reduced by increasing the durability and, therefore, extending its lifetime. To showcase the proposed method's capability, two design variables have been chosen; the height of the beam and the ratio of ordinary Portland cement (OPC) to its replacement binder ground granulated blast furnace slag (BFS), a by-product of the iron industry. In addition to the static design according to the standard, the problem is extended to include a key performance indicator (KPI) related to the production process in a prefabrication factory that defines the time after which the removal of the formwork can be performed. To approximate this, the point in time when the beam can bear its own weight has been chosen a criterion. Reducing this time equates to being able to produce more parts with the same formwork.

2. Methods

2.1. Design approaches

The conventional method of designing reinforced concrete structures is depicted in Figure 2. The structural engineer starts by choosing a suitable material (e.g., strength class C40/50) and designs the structure, including the geometry (e.g., the height of a beam) and the reinforcement. In the second step, this design is handed over to the material engineer with the constraint that the material properties assumed by the structural engineer have to be met. This lack of coordination strongly restricts the set of potential solutions since structural design and concrete mix design are strongly coupled; for example, a lower strength can be compensated with a larger beam height.

An alternative design workflow is illustrated in Figure 3, which entails inverting the classical design pipeline. The material composition is the input to the material engineer who predicts the corresponding mechanical properties of the material. This includes KPIs related to the material, for example, viscosity/slump test, or simply the water/cement ratio. In a second step, a structural analysis is performed with the material properties as input. This step outputs the structural KPIs such as the load-bearing capacity, the expected lifetime (for a structure susceptible to fatigue), or the GWP of the complete structure. These two (coupled) modules are used within an optimization procedure to estimate the optimal set of input parameters (both on the material level and on the structural level). One of the KPIs is chosen as the objective function (e.g., GWP) and others as constraints (e.g., load-bearing capacity larger than the load, cement content larger than a threshold, viscosity according to the slump test within a certain interval). Note that in order to use such an inverse-design procedure, the

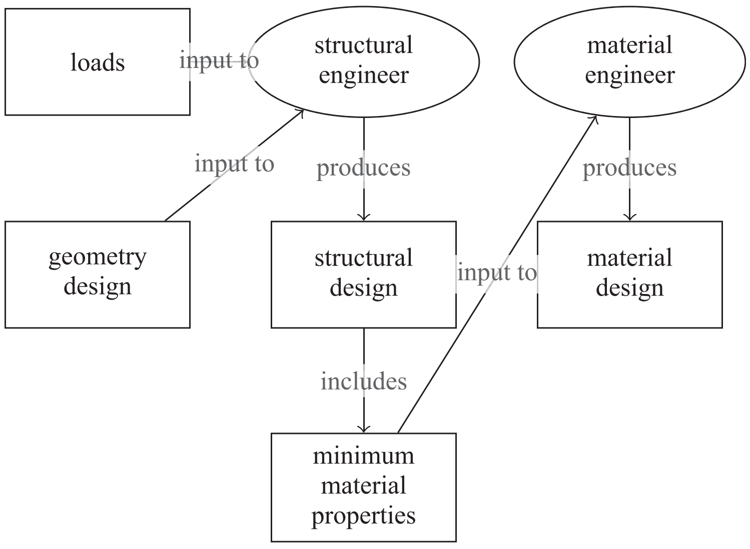


Figure 2. *Classical design approach, where the required minimum material properties are defined by the structural engineer which is then passed to the material engineer.*

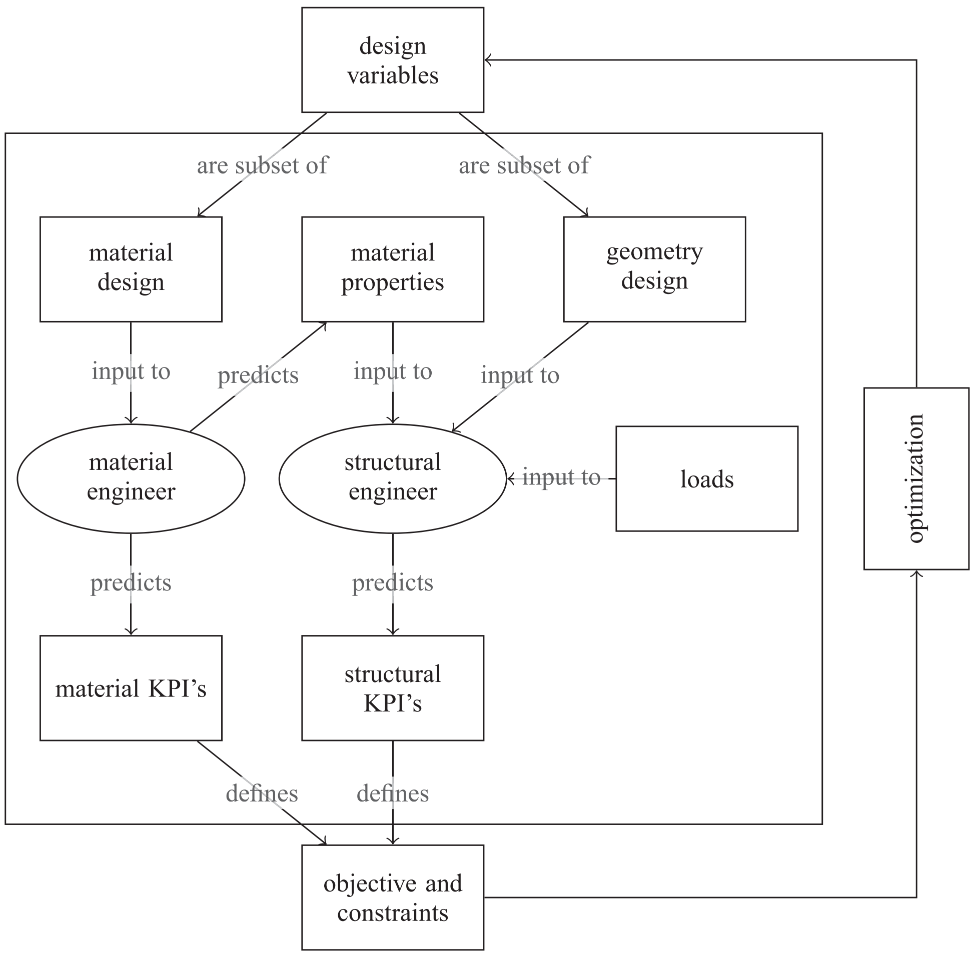


Figure 3. *Proposed design approach to cast material and structural design into a forward model that is then integrated into a holistic optimization approach.*

forward modeling workflow needs to be automated and subsequently the information needs to be efficiently back-propagated.

This article aims to present the proposed methodological framework as well as illustrate its capabilities in the design of a precast concrete element with the objective of reducing the GWP. The constraints employed are related to the structural performance after 28 days as well as the maximum time of demolding after 10 hours. The design/optimization variables are, on the structural level, the height of the beam, and on the material level, the composition of the binder as a mixture of Portland cement and slag. The complete workflow is illustrated in Figure 4.

2.2. Workflow for predicting key performance indicators

The workflow consists of four major steps. In the first step, the cement composition (blended cement and slag) defined in the mix composition is used to predict the mechanical properties of the cement paste. This is done using a data-driven approach as discussed in Section 2.4. In the second step, homogenization is used in order to compute the effective, concrete properties based on cement paste and aggregate data. An analytical function is applied for the homogenization based on the Mori–Tanaka scheme (Mori and Tanaka, 1973). The third step involves a multi-physics, FE model with two complex constitutive models —a hydration model, which computes the evolution of the degree of hydration (DOH), considering the local temperature and the heat released during the reaction, and a mechanical model, which simulates the temporal evolution of the mechanical properties assuming that those depend on the DOH. The fourth and last model is based on a design code to estimate the amount of reinforcement and predict the load-bearing

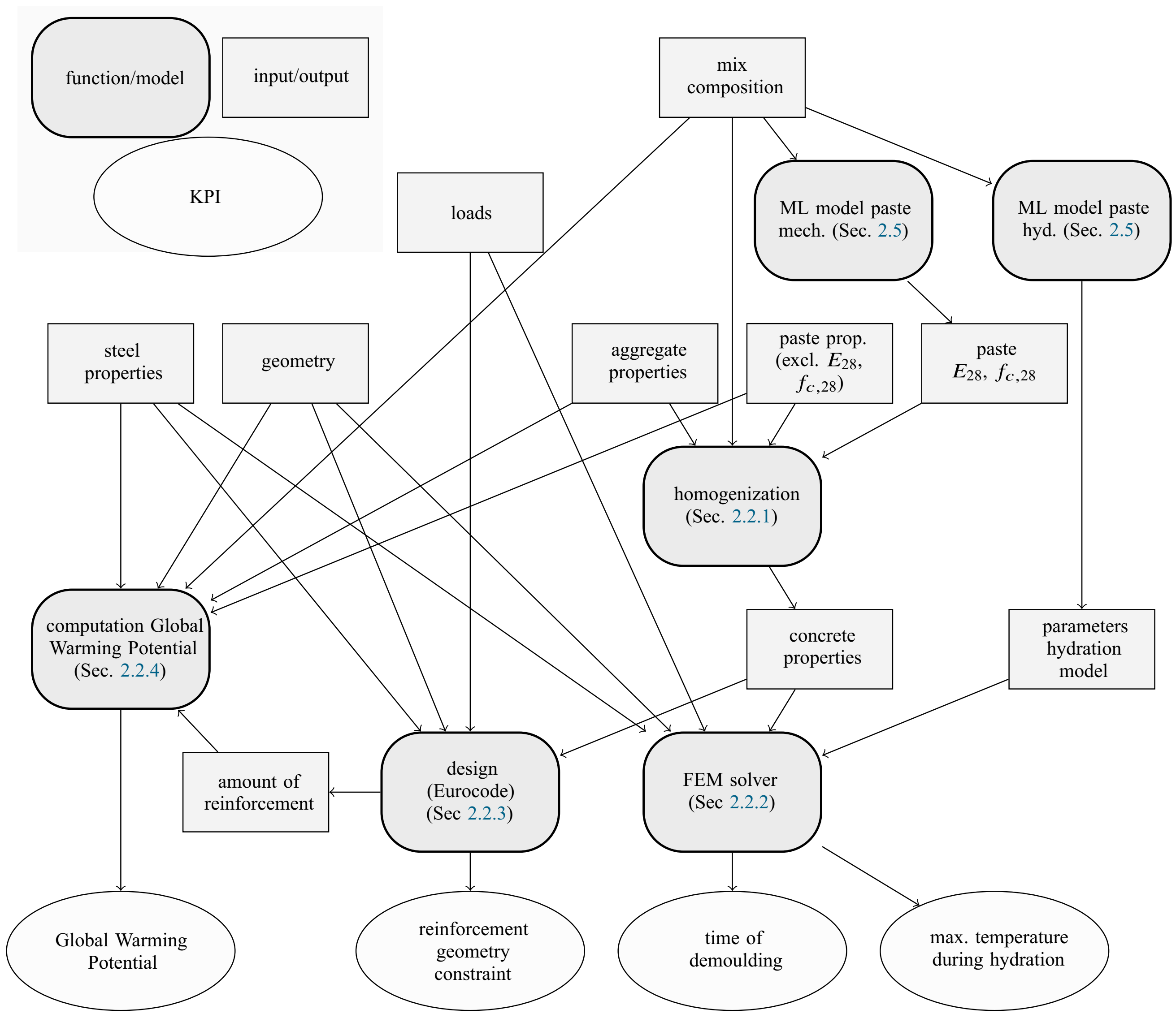

function/model
input/output
KPI
mix composition
loads
ML model paste mech. (Sec. 2.5)
ML model paste hyd. (Sec. 2.5)
steel properties
geometry
aggregate properties
paste prop. (excl. E_{28}, $f_{c,28}$)
paste E_{28}, $f_{c,28}$
homogenization (Sec. 2.2.1)
computation Global Warming Potential (Sec. 2.2.4)
concrete properties
parameters hydration model
amount of reinforcement
design (Eurocode) (Sec 2.2.3)
FEM solver (Sec 2.2.2)
Global Warming Potential
reinforcement geometry constraint
time of demoulding
max. temperature during hydration

Figure 4. *Workflow to compute key performance indicators from input parameters.*

capacity after 28 days. Subsequent sections will provide insights into how these models function within the optimization framework.

2.2.1. Homogenized concrete parameters

Experimental data for estimating the compressive strength are obtained from concrete specimens measuring the homogenized response of cement paste and aggregates. The mechanical properties of aggregates are known, whereas the cement paste properties have to be inversely estimated. The calorimetry is directly performed for cement paste.

In order to relate macroscopic mechanical properties to the individual constituents (cement paste and aggregates), an analytical homogenization procedure is used. The homogenized effective concrete properties are the Young modulus E, the Poisson ratio ν, the compressive strength f_c, the density ρ, the thermal conductivity χ, the heat capacity C, and the total heat release Q_∞. Depending on the physical meaning, these properties need slightly different methods to estimate the effective concrete properties. The elastic, isotropic properties E and ν of the concrete are approximated using the Mori–Tanaka homogenization scheme (Mori and Tanaka, 1973). The method assumes spherical inclusions in an infinite matrix and considers the interactions of multiple inclusions. Details are given in Appendix A.1.

The estimation of the concrete compressive strength $f_{c,\mathrm{eff}}$ follows the ideas of Nežerka et al. (2018). The premise is that a failure in the cement paste will cause the concrete to crack. The approach is based on

Table 1. *Properties of the cement paste and aggregates used in subsequent sensitivity studies*

Phase	E[Pa]	ν[−]	f_c[Pa]	ρ[kg/m^3]	χ[J/kgK]	C[W/mK]	Q_∞[J/kg]
Paste	30e9	0.2	30e6	2,400	870	1.8	250,000
Aggr.	25e9	0.3	—	2,600	840	0.8	0

two main assumptions. First, the Mori–Tanaka method is used to estimate the average stress within the matrix material $\sigma^{(m)}$. Second, the von Mises failure criterion of the average matrix stress is used to estimate the uniaxial compressive strength (see Appendix A.1.1).

Table 1 gives an overview of the material properties of the constituents used in the subsequent sensitivity studies. The effective properties as a function of the aggregate content are plotted in Figure 5. Note that both extremes (0 [pure cement] and 1 [only aggregates]) are purely theoretical.

For the considered example, the relations are close to linear. This can change when the difference between the matrix and the inclusion properties is more pronounced or more complex micromechanical mechanisms are incorporated, such as air pores or the interfacial transition zone. Though not done in this article, these could be considered within the chosen homogenization scheme by adding additional phases (cf. Nežerka and Zeman, 2012). Homogenization of the thermal conductivity is also based on the Mori–Tanaka method, following the ideas of Stránský et al. (2011) with details given in Appendix A.1.2. The density ρ, the heat capacity C, and the total heat release Q_∞ can be directly computed based on their volume average. As an example for the volume-averaged quantities, the heat release is shown in Figure 5 as it exemplifies the expected linear relation of the volume average as well as the zero heat output of a theoretical pure aggregate.

2.2.2. Hydration and evolution of mechanical properties

Due to a chemical reaction (hydration) of the binder with water, calcium-silicate hydrates form that lead to a temporal evolution of concrete strength and stiffness. The reaction is exothermal and the kinetics are sensitive to the temperature. The primary model simulates the hydration process and computes the temperature field T and the DOH α (see Eqs. (B1) and (B2) in Appendix B). The latter characterizes the DOH that condenses the complex chemical reactions into a single scalar variable. The thermal model depends on three material properties: the effective thermal conductivity λ, the specific heat capacity C, and the heat release $\frac{\partial Q}{\partial t}$. The latter is governed by the hydration model, characterized by six parameters: $B_1, B_2, \eta, T_{\text{ref}}, E_a$, and $\alpha_{\max}$. The first three B_1, B_2, and η are parameters characterizing the shape of the evolution of the heat release. T_{ref} is the reference temperature for which the first three parameters are calibrated. (Based on the difference between the actual and the reference temperature, the heat released is scaled.) The sensitivity to the temperature is characterized by the activation energy E_a. $\alpha_{\max}$ is the maximum DOH that can be reached. Following Mills (1966), the maximum DOH is estimated based on the water to binder ratio r_{wc}, as $\alpha_{\max} = \frac{1.031\, r_{\text{wc}}}{0.194 + r_{\text{wc}}}$.

By assuming the DOH to be the fraction of the currently released heat with respect to its theoretical potential Q_∞, the current DOH is estimated as $\alpha(t) = \frac{Q(t)}{Q_\infty}$. As the potential heat release is also difficult to

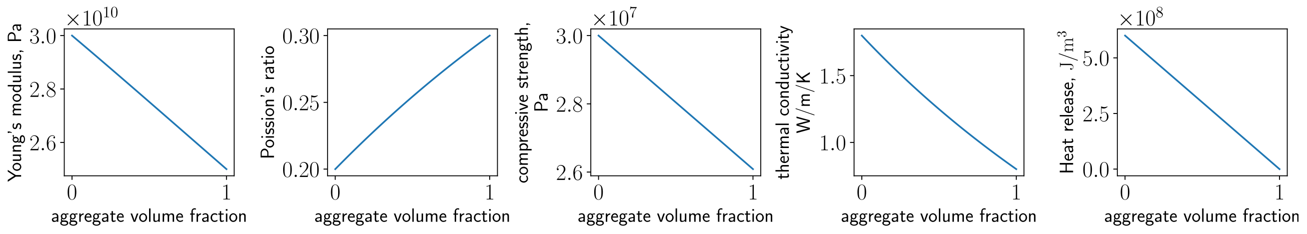


Figure 5. *Influence of aggregate ratio on effective concrete properties.*

measure as it takes a long time to fully hydrate and will only do so under perfect conditions, we identify it as an additional parameter in the model parameter estimation. For a detailed model description, see Appendix B. In addition to influencing the reaction speed, the computed temperature is used to verify that the maximum temperature during hydration does not exceed a limit of $T_{\text{limit}} = 60\,^\circ\text{C}$. Above a certain temperature, the hydration reaction changes (e.g., secondary ettringite formation) and, additionally, volumetric changes in the cooling phase correlate with cracking and reduced mechanical properties. The maximum temperature is implemented as a constraint for the optimization problem (see Eq. (B19)).

The evolution of the Young modulus E of a linear-elastic material model is modeled as a function of the DOH (details in Eq. (B17)). In a similar way, the compressive strength evolution is computed (see Eq. (B15)), which is utilized to determine a failure criterion based on the computed local stresses (Eq. (B20)) related to the time when the formwork can be removed. For a detailed description of the parameter evolution as a function of the DOH, see Appendix B.2. Figure 6 shows the influence of the different parameters. In addition to the formulations given in Carette and Staquet (2016) which depend on a theoretical value of parameters for fully hydrated concrete at $\alpha = 1$, this work reformulates the equations, to depend on the 28 day values E_{28} and f_{c28} as well as the corresponding α_{28} which is obtained via a simulation. This allows us to directly use the experimental values as input. In Figure 6, α_{28} is set to 0.8.

2.2.3. Beam design according to EC2

The design of the reinforcement and the computation of the load-bearing capacity is performed based on DIN EN 1992-1-1 (2011) according to Eq. (C7) with a detailed explanation in the Appendix C. To ensure that the design is realistic, the continuous cross section is transformed into a discrete number of bars with a diameter chosen from a list. This is visible in Figure 7 by the stepwise increase in cross sections. The admissible results are restricted by two constraints. One is coming from a minimal required compressive strength (Eq. (C8)), visualized as a dashed line. The other, based on the available space to place bars with admissible spacing (Eq. (C13)), visualized as the dotted line. Further details on the computation are given in Appendix C. A sensitivity study for the mutual interaction and the constraints is visualized in Figure 7. The parameters for the sensitivity study are given in Table D1 in Appendix D.

2.2.4. Computation of GWP

The computation of the GWP is performed by multiplying the volume content of each individual material by its specific GWP. The values used in this study are extracted from Braga et al. (2017) and listed in Table 2.

We note that the question of what exactly to include in the GWP computation is largely open. For example, the transport of materials, while non-negligible, is difficult in general to include. Furthermore, there are always local conditions (e.g., the GWP of the energy sources used in the cement production

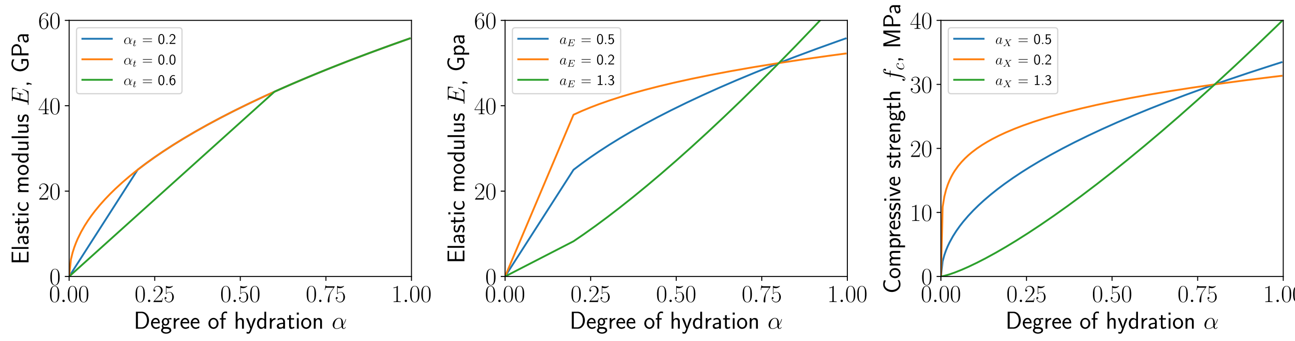


Figure 6. *Influence of parameters α_t, a_E, and a_{f_c} on the evolution the Young modulus and the compressive strength with respect to the degree of hydration α. Parameters: $E_{28} = 50\,\text{GPa}$, $a_E = 0.5$, $\alpha_t = 0.2$, $a_{f_c} = 0.5$, $f_{c28} = 30\,\text{N/mm}^2$, $\alpha_{28} = 0.8$.*

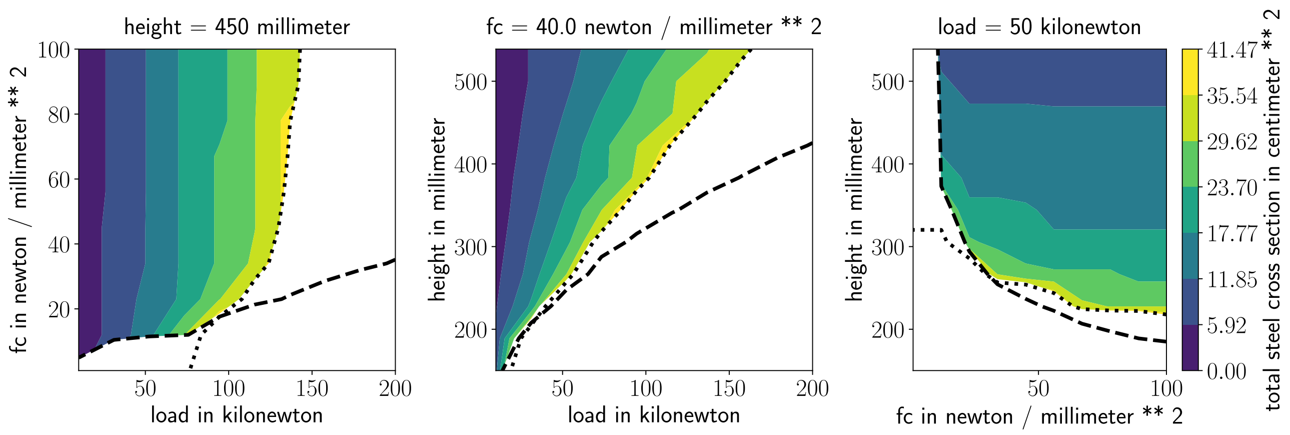


Figure 7. *Influence of beam height, concrete compressive strength, and load in the center of the beam on the required steel. The dashed lines represent the minimum compressive strength constraint (Eq. (C8)), and the dotted lines represent the geometrical constraint from the spacing of the bars (Eq. (C13)).*

Table 2. *Specific global warming potential of the raw materials used in the design*

Material	GWP $\left[\frac{\mathrm{kg}_{\mathrm{CO_2eq}}}{\mathrm{m}^3}\right]$
Portland cement	0.95
Slag	0.18
Aggregates	0.025
Water	0.000133
Steel	1.42

depends on the amount of green energy in that country). In addition, the time span (complete life cycle analysis vs. production) is a point of debate and finally the usage of by-products (slag is currently a by-product of steel manufacturing and thus its GWP is considered to be small). There are currently efforts, both at the national and international levels, to standardize the computation of the GWP or similar quantities. Once these are available, they can be readily integrated into the proposed approach. Such a standardized computation of the GWP can lead either to taxing GWP or to introducing a sustainability limit state, though this is an ongoing discussion in standardization committees.

2.3. Experimental data

This section describes the data used for learning the missing (probabilistic) links (detailed in Section 2.5) between the slag–binder mass ratio r_{sb} and physical model parameters. The slag–binder mass ratio r_{sb} is the mass ratio between the amount of BFS and the binder (sum of BFS and OPC). The data are sourced from Gruyaert (2011). In particular, we are concerned about the parameter estimation for the concrete homogenization discussed in Section 2.2.1 and the hydration model in Section 2.2.2.

For concrete homogenization, six different tests for varying ratios $r_{\mathrm{sb}} = \{0.0, 0.15, 0.3, 0.5, 0.7, 0.85\}$ are available for the concrete compressive strength f_c after 28 days. For the concrete hydration, we utilize isothermal calorimetry data at 20 °C. We have temporal evolution data of cumulative heat of hydration $\hat{\boldsymbol{Q}}$ for four different values of $r_{\mathrm{sb}} = \{0.0, 0.30, 0.50, 0.85\}$. For details on other material parameters and phenomenological values used to obtain the data, the reader is directed to Gruyaert (2011).

2.3.1. Young's modulus E based on f_c

The dataset does not encompass information about the Young modulus. Given its significance for the FEM simulation, we resort to a phenomenological approximation derived from ACI Committee

363 (2010). This approximation relies on the compressive strength f_c and the density ρ to estimate the Young modulus

$$E = 3{,}320\sqrt{f_c} + 6{,}895\left(\frac{\rho}{2{,}320}\right)^{1.5}, \tag{1}$$

with ρ in kilogram per cubic meter and f_c and E in megapascals.

2.4. Model learning and optimization

The workflow illustrated in Figure 4, which builds the link between the parameters relevant to the concrete mix design and the KPIs involving the environmental impact and the structural performance, can be represented in terms of the probabilistic graph (Koller and Friedman, 2009) shown in Figure 8. As discussed in the Introduction (Section 1), the goal of the present study is to find the value of the design variables $\boldsymbol{x}$ (concrete mix design and beam geometry) which minimizes the objective $\mathcal{O}$ (environmental impact), while satisfying a given set of constraints $\mathcal{C}_i$ (beam design criterion, structural performance, etc.). This necessitates forward and backward information flow in the presented graph. The forward information flow is necessary to compute the KPIs for given values of the design variables and the backward information is essentially the sensitivities of the objective and the constraints with respect to the design variables that enable gradient-based optimization. Establishing the information flow poses challenges, which we attempt to tackle with the methods proposed as follows:

- *Data-based model learning:* The physics-based models discussed in Sections 2.2.1 and 2.2.2 are used to compute various KPIs (discussed in Figure 8). These depend on some model parameters denoted by b which are unobserved (latent) in the experiments performed. The model parameters need not only be inferred on the basis of experimental data but also their dependence on the design variables $\boldsymbol{x}$ is required in order to be integrated into the optimization framework. In addition, the noise in the data (aleatoric) or the incompleteness of data (epistemic) introduces uncertainty. To this end, we propose learning *probabilistic* links by employing experimental data as discussed in detail in Section 2.5.

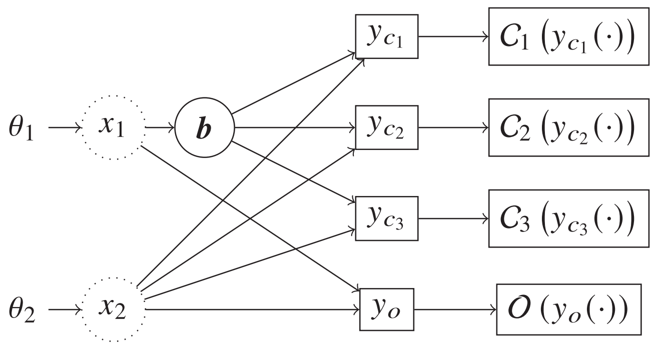

θ_1
x_1
b
θ_2
x_2
y_{c_1}
y_{c_2}
y_{c_3}
y_o
$\mathcal{C}_1\left(y_{c_1}(\cdot)\right)$
$\mathcal{C}_2\left(y_{c_2}(\cdot)\right)$
$\mathcal{C}_3\left(y_{c_3}(\cdot)\right)$
$\mathcal{O}\left(y_o(\cdot)\right)$

Figure 8. *Stochastic computational graph for the constrained optimization problem for performance-based concrete design: The circles represent stochastic nodes and rectangles deterministic nodes. The design variables are denoted by $\boldsymbol{x} = (x_1, x_2)$. The vector b represents the unobserved model parameters which are needed in order to link the key performance indicators (KPIs) $\boldsymbol{y} = \left(y_o, \{y_{c_i}\}_{i=1}^{I}\right)$ with the design variables $\boldsymbol{x}$. Here, y_o represents the model output appearing in the optimization objective and y_{c_i} represents the model output appearing in the* i*th constraint. The objective function is given by $\mathcal{O}$ and the* i*th constraint by $\mathcal{C}_i$. They are not differentiable with respect to x_1, x_2. (Hence, x_1 and x_2 are dotted.) The variables $\boldsymbol{\theta}$ are auxiliary and are used in the context of Variational Optimization discussed in Section 2.6.2. Several other deterministic nodes are present between the random variables **b** and the KPIs **y**, but they are omitted for brevity. The physical meaning of the variables used is detailed in Table 3.*

Table 3. *Physical meaning of the variables used in Figure 8*

Variables	Physical meaning
x_1	Mass ratio of blast furnace slag and ordinary Portland cement r_{sb}
x_2	Beam height h
$\boldsymbol{b}$	Vector of the input, model parameters to the homogenization and hydration model (Sections 2.2.1 and 2.2.2, respectively)
y_{c_1}	Required steel reinforcement area (Eq. (C7))
$\mathcal{C}_1(y_{c_1}(\cdot))$	Beam design constraint (Eq. (C14))
y_{c_2}	Max. temperature reached (Appendix B.3)
$\mathcal{C}_2(y_{c_2}(\cdot))$	Temperature constraint (Eq. (B19))
y_{c_3}	Time of demolding (Appendix B.3)
$\mathcal{C}_3(y_{c_3}(\cdot))$	Time constraint based on yield strength (Eq. (B20))
y_o	The GWP of the beam (Section 2.2.4)
$\mathcal{O}_o(y_o(\cdot))$	Objective corresponding to the beam GWP

- *Optimization under uncertainty:* The aforementioned uncertainties as well as additional randomness that might be present in the associated links necessitate reformulating the optimization problem (i.e., objectives/constraints) as one of OUU. In turn, this gives rise to new challenges in order to compute the needed derivatives of the KPIs with respect to the design variables which are discussed in Section 2.6.

2.5. Probabilistic links based on data and physical models

This section deals with learning a (probabilistic) model linking the design variables $\boldsymbol{x}$ and the input parameters of the physics-based models, that is, concrete hydration and concrete homogenization. A graphical representation is contained in Figure 9. Therein, $\left\{\hat{\boldsymbol{x}}^{(i)}, \hat{\boldsymbol{z}}^{(i)}\right\}_{i=1}^{N}$ denote the observed data pairs and $\boldsymbol{b}$ denotes a vector of unknown and unobserved parameters of the physics-based models and $\boldsymbol{z}(\boldsymbol{b})$ the model outputs. The latter relate to an experimental observation $\hat{\boldsymbol{z}}^{(i)}$ as $\hat{\boldsymbol{z}}^{(i)} = \boldsymbol{z}\left(\boldsymbol{b}^{(i)}\right) + \text{noise}$, which gives rise to a likelihood $p\left(\hat{\boldsymbol{z}}^{(i)} | \boldsymbol{z}\left(\boldsymbol{b}^{(i)}\right)\right)$. We further postulate a probabilistic relation between $\hat{\boldsymbol{x}}$ and $\boldsymbol{b}$ that is expressed by the conditional $p(\boldsymbol{b}|\boldsymbol{x};\varphi)$, which depends on unknown parameters $\boldsymbol{\varphi}$. The physical meaning of the aforementioned variables and model links, as well as of the relevant data, is presented in Table 4. The elements introduced above suggest a Bayesian formulation that can quantify inferential uncertainties in the unknown parameters and propagate it in the model predictions (Koutsourelakis et al., 2016), as detailed in the next section.

2.5.1. Expectation–maximization

Given N data pairs $\mathcal{D}_N = \left\{\hat{\boldsymbol{x}}^{(i)}, \hat{\boldsymbol{z}}^{(i)}\right\}_{i=1}^{N}$ consisting of different concrete mixes and corresponding outputs, we would like to infer the corresponding $\boldsymbol{b}^{(i)}$, but more importantly the relation between $\boldsymbol{x}$ and $\boldsymbol{b}$ which would be of relevance for downstream, optimization tasks discussed in Section 2.6.

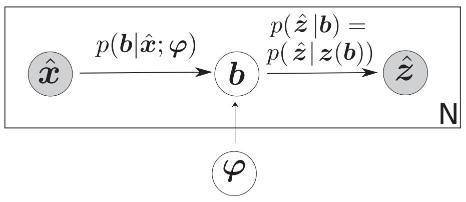

$p(\boldsymbol{b}|\hat{\boldsymbol{x}};\varphi)$
$p(\hat{\boldsymbol{z}}|\boldsymbol{b}) = p(\hat{\boldsymbol{z}}|\boldsymbol{z}(\boldsymbol{b}))$
$\hat{\boldsymbol{x}}$
$\boldsymbol{b}$
$\hat{\boldsymbol{z}}$
φ
N

Figure 9. *Probabilistic graph for the data and physical model-based model learning. The shaded nodes are the observed and unshaded are the unobserved (latent) nodes.*

Table 4. *Physical meaning of the variables/links used in Figure 9*

$\boldsymbol{b}$ (model input)	$\hat{z}$ (observed data)	$z(\boldsymbol{b})$ (physics-based model)
Hydration model input parameters (Section 2.2.2) $\boldsymbol{b} = \left[B_1, B_2, \eta, Q_{\text{pot}}\right]$	Heat of hydration $\boldsymbol{Q}$	Concrete hydration model
Cement paste compressive strength ($f_{c,\text{paste}}$), cement paste Young's Modulus (E_{paste}) (Section 2.2.1)	Concrete compressive strength (f_c), concrete Young's Modulus (E_c)	Concrete homogenization model

Note. $\hat{x}$ is the slag–binder mass ratio r_{sb} for the all the cases presented above.

We postulate a probabilistic relation between $\boldsymbol{x}$ and $\boldsymbol{b}$ in the form of a conditional density $p(\boldsymbol{b}|\boldsymbol{x};\boldsymbol{\varphi})$ parametrized by $\boldsymbol{\varphi}$. For example,

$$p(\boldsymbol{b}|\boldsymbol{x};\boldsymbol{\varphi}) = \mathcal{N}\left(\boldsymbol{b}|\boldsymbol{f}_\varphi(\boldsymbol{x}),\, \boldsymbol{S}_\varphi(\boldsymbol{x})\right), \tag{2}$$

where $\boldsymbol{f}_\varphi(\boldsymbol{x})$ represents a fully connected, feed-forward neural network parametrized by $\boldsymbol{w}$ (further details discussed in Section 3) and $\boldsymbol{S}_\varphi(\boldsymbol{x}) = \boldsymbol{L}\boldsymbol{L}^T$ denotes the covariance matrix where $\boldsymbol{L}$ is lower-triangular. Hence, the parameters $\boldsymbol{\varphi}$ to be learned correspond to $\boldsymbol{\varphi} = \{\boldsymbol{w}, \boldsymbol{L}\}$. We assume that the observations $\hat{z}^{(i)}$ are contaminated with Gaussian noise, which gives rise to the likelihood:

$$p\left(\hat{z}^{(i)}|z\left(\boldsymbol{b}^{(i)}\right)\right) = \mathcal{N}\left(\hat{z}^{(i)}|z\left(\boldsymbol{b}^{(i)}\right),\, \boldsymbol{\Sigma}_\ell\right). \tag{3}$$

The covariance $\boldsymbol{\Sigma}_\ell$ depends on the data used and is discussed in Section 3.

Given Eqs. (2) and (3), one can observe that $\boldsymbol{b}^{(i)}$ (i.e., the unobserved model inputs for each concrete mix i) and $\boldsymbol{\varphi}$ would need to be inferred simultaneously. In the following, we obtain point estimates $\boldsymbol{\varphi}^*$ for $\boldsymbol{\varphi}$, by maximizing the marginal log-likelihood $p(\mathcal{D}_N|\boldsymbol{\varphi})$ (also known as log-evidence), that is, the probability that the observed data arose from the model postulated. Hence, we get

$$\boldsymbol{\varphi}^* = \arg\max_{\boldsymbol{\varphi}} \log p(\mathcal{D}_N|\boldsymbol{\varphi}). \tag{4}$$

As this is analytically intractable, we propose employing variational Bayesian expectation–maximization (VB-EM) (Beal and Ghahramani, 2003) according to which a lower bound $\mathcal{F}$ to the log-evidence (called evidence lower bound [ELBO]) is constructed with the help of auxiliary densities $h_i\left(\boldsymbol{b}^{(i)}\right)$ on the unobserved variables $\boldsymbol{b}^{(i)}$:

$$\log p(\mathcal{D}_N|\varphi) \geq \underbrace{\sum_{i=1}^{N} \mathbb{E}_{h_i\left(\boldsymbol{b}^{(i)}\right)}\left[\log \frac{p\left(\hat{z}^{(i)}|z\left(\boldsymbol{b}^{(i)}\right)\right)p\left(\boldsymbol{b}^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\varphi}\right)}{h_i\left(\boldsymbol{b}^{(i)}\right)}\right]}_{=\mathcal{F}(h_{1:N},\boldsymbol{\varphi})}, \tag{5}$$

where $\mathbb{E}_{h_i\left(\boldsymbol{b}^{(i)}\right)}[\cdot]$ denote the expectation with respect to the auxiliary densities $h_i\left(\boldsymbol{b}^{(i)}\right)$ on the unobserved variables $\boldsymbol{b}^{(i)}$. Eq. (5) suggests the following iterative scheme where one alternates between the steps:

- **E-step**: Fix $\boldsymbol{\varphi}$ and maximize $\mathcal{F}$ with respect to $h_i\left(\boldsymbol{b}^{(i)}\right)$. It can be readily shown (Bishop and Nasrabadi, 2006) that optimality is achieved by the conditional posterior, that is,

$$h_i^{opt}\left(\boldsymbol{b}^{(i)}\right) = p\left(\boldsymbol{b}^{(i)}|\mathcal{D}_N, \boldsymbol{\varphi}\right) \propto p\left(\hat{z}^{(i)}|\boldsymbol{b}^{(i)}\right)p\left(\boldsymbol{b}^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{\varphi}\right), \tag{6}$$

which makes the inequality in Eq. (5) tight. Since the likelihood is not tractable as it involves a physics-based solver, we have used Markov chain Monte Carlo (MCMC) to sample from the conditional posterior (see Section 3).

- **M-step**: Given $\{h_i(\boldsymbol{b}^{(i)})\}_{i=1}^N$, maximize $\mathscr{F}$ with respect to $\boldsymbol{\varphi}$.

$$\boldsymbol{\varphi}^{n+1} = \arg\max_{\boldsymbol{\varphi}} \mathscr{F}(h_{1:N}, \boldsymbol{\varphi}^n). \tag{7}$$

This requires derivatives of $\mathscr{F}$, that is,

$$\frac{\partial \mathscr{F}}{\partial \boldsymbol{\varphi}} = \sum_{i=1}^{N} \mathbb{E}_{h_i}\left[\frac{\partial \log p(\boldsymbol{b}^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\varphi})}{\partial \boldsymbol{\varphi}}\right]. \tag{8}$$

Given the MCMC samples $\{\boldsymbol{b}_m^{(i)}\}_{m=1}^M$ from the E-step, these can be approximated as

$$\frac{\partial \mathscr{F}}{\partial \varphi} \approx \sum_{i=1}^{N} \frac{1}{M} \sum_{m=1}^{M} \frac{\partial \log p(\boldsymbol{b}_m^{(i)}|\boldsymbol{x}^{(i)};\varphi)}{\partial \varphi}. \tag{9}$$

Due to the Monte Carlo noise in these estimates, a stochastic gradient ascent algorithm is utilized. In particular, the ADAM optimizer (Kingma and Ba, 2014) was used from the `PyTorch` (Paszke et al., 2019) library to capitalize on its auto-differentiation capabilities.

The major elements of the method are summarized in Algorithm 1. We note here that training complexity grows linearly with the number of training samples N due to the densities h_i associated with each data point (**for**-loop of Algorithm 1), but this can be embarrassingly parallelizable.[1]

Algorithm 1 Data-based model learning.

```
1: Input: Data $\mathcal{D}_N = \{\hat{\boldsymbol{x}}^{(i)}, \hat{\boldsymbol{z}}^{(i)}\}_{i=1}^N$, model form $p(\boldsymbol{b}|\boldsymbol{x};\boldsymbol{\varphi})$, likelihood noise $\boldsymbol{\Sigma}_l$, $n = 0$
2: Output: Learned parameter $\boldsymbol{\varphi}^*$
3: Initialize the parameters $\boldsymbol{\varphi}$
4: while ELBO not converged do
       Expectation Step (E-step):
5:     for $i = 1$ to $N$ do
6:         MCMC to get the posterior probability $p(\boldsymbol{b}^{(i)}|\mathcal{D}_N, \boldsymbol{\varphi}^n)$ using current $\boldsymbol{\varphi}^n$    ▷ Eq. (6)
7:     end for
       Maximization Step (M-step):
8:     Monte Carlo gradient estimate    ▷ Eq. (9)
9:     $\boldsymbol{\varphi}^{n+1} = \arg\max_{\boldsymbol{\varphi}} \mathscr{F}(h_{1:N}, \boldsymbol{\varphi}^n)$
10:    $n \leftarrow n+1$
11: end while
```

Model Predictions: The VB-EM based model learning scheme discussed above can be carried out in an *offline phase.* Once the model is learned, we are interested in the proposed model's ability to produce probabilistic predictions (*online stage*) that reflect the various sources of uncertainty discussed previously. For learnt parameters $\boldsymbol{\varphi}^*$, the predictive posterior density $p_{\text{pred}}(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\varphi}^*)$ on the solution vector $\boldsymbol{z}$ of a physical model is as follows:

$$\begin{aligned} p_{\text{pred}}(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\varphi}^*) &= \int p(\boldsymbol{z}, \boldsymbol{b}|\mathcal{D}, \boldsymbol{\varphi}^*) \mathrm{d}\boldsymbol{b} \\ &= \int p(\boldsymbol{z}|\boldsymbol{b}) p(\boldsymbol{b}|\mathcal{D}, \boldsymbol{\varphi}^*) \mathrm{d}\boldsymbol{b} \end{aligned} \tag{10}$$

[1] Embarrassingly parallelizable problems are problems that can be separated into parallel tasks without the need of any interaction between the simulations performed at different processing units (Herlihy et al., 2020).

$$\approx \frac{1}{K}\sum_{k=1}^{K} z\left(\boldsymbol{b}^{(k)}\right). \tag{11}$$

The second of the densities is the conditional (Eq. (2)) substituted with the learned $\boldsymbol{\varphi}^*$ and the first of the densities is simply a Dirac-delta that corresponds to the solution of the physical model, that is, $z(\boldsymbol{b})$. The intractable integral can be approximated by Monte Carlo using K samples of $\boldsymbol{b}$ drawn from $p(\boldsymbol{b}|\mathcal{D},\boldsymbol{\varphi}^*)$.

2.6. *Optimization under uncertainty*

With the relevant missing links identified as detailed in the previous section, the optimization can be performed on the basis of Figure 8. We seek to optimize the objective function $\mathcal{O}$ subject to constraints $\mathcal{C} = (\mathcal{C}_1,\ldots,\mathcal{C}_I)$ that are dependent on uncertain parameters $\boldsymbol{b}$, which in turn are dependent on the design variables $\boldsymbol{x}$. In this setting, the general parameter-dependent nonlinear constrained optimization problem can be stated as

$$\begin{aligned} &\min_{\boldsymbol{x}} \mathcal{O}(y_o(\boldsymbol{x},\boldsymbol{b})), \\ &\text{s.t. } \mathcal{C}_i\left(y_{c_i}(\boldsymbol{x},\boldsymbol{b})\right) \le 0,\ \forall i \in \{1,\ldots,I\}, \end{aligned} \tag{12}$$

where $\boldsymbol{x}$ is a d-dimensional vector of design variables and $\mathbf{b}$ are the model parameter discussed in the previous section. It can be observed that the optimization problem is nontrivial because of three main reasons: (a) the presence of the constraints (Section 2.6.1), (b) the presence of random variables $\boldsymbol{b}$ in the objective and the constraint(s) (Section 2.6.1), and (c) non-differentiability of y_o, y_{c_i} and therefore of $\mathcal{O}$ and $\mathcal{C}_i$.

2.6.1. *Handling stochasticity and constraints*

Since the solution of Eq. (12) depends on the random variables $\boldsymbol{b}$, the objective and constraints are random variables as well and we have to take their random variability into account. We do this by reverting to a robust optimization problem (Ben-Tal and Nemirovski, 1999; Bertsimas et al., 2011), with expected values denoted by $\mathbb{E}[\cdot]$ being the robustness measure to integrate out the uncertainties. In this manner, the optimization problem in Eq. (12) is reformulated as

$$\begin{aligned} &\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}}[\mathcal{O}(y_o(\boldsymbol{x},\boldsymbol{b}))], \\ &\text{s.t. } \mathbb{E}_{\mathbf{b}}\left[\mathcal{C}_i\left(y_{c_i}(\boldsymbol{x},\boldsymbol{b})\right)\right] \le 0,\ \forall i \in \{1,\ldots,I\}. \end{aligned} \tag{13}$$

The expected objective value will yield a design that performs best on average, while the reformulated constraints imply feasibility on average.

One can cast this constrained problem to an unconstrained one using penalty-based methods (Nocedal and Wright, 1999; Wang and Spall, 2003). In particular, we define an augmented objective function $\mathcal{L}$ as follows:

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) = \mathcal{O}(y_o(\boldsymbol{x},\boldsymbol{b})) + \sum_i \lambda_i \max\left(\mathcal{C}_i\left(y_{c_i}(\boldsymbol{x},\boldsymbol{b})\right),0\right), \tag{14}$$

where $\lambda_i > 0$ is the penalty parameter for the ith constraint. The larger the λ_i's are, the more strictly the constraints are enforced. Incorporating the augmented objective (Eq. (14)) in the reformulated optimization problem (Eq. (13)), one can arrive at the following penalized objective:

$$\mathbb{E}_{\boldsymbol{b}}[\mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})] = \int \mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})p(\boldsymbol{b}|\boldsymbol{x},\boldsymbol{\varphi})\mathrm{d}\boldsymbol{b}, \tag{15}$$

leading to the following equivalent, unconstrained optimization problem:

$$\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}}[\mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})]. \tag{16}$$

The expectation above is approximated by Monte Carlo which induces noise and necessitates the use of stochastic optimization methods (discussed in detail in the sequel). Furthermore, we propose to alleviate the dependence on the penalty parameters λ by using the sequential unconstrained minimization technique algorithm (Fiacco and McCormick, 1990), which has been shown to work with nonlinear constraints (Liuzzi et al., 2010). The algorithm considers a strictly increasing sequence $\{\boldsymbol{\lambda}_n\}$ with $\boldsymbol{\lambda}_n \to \infty$. Fiacco and McCormick (1990) proved that when $\boldsymbol{\lambda}_n \to \infty$, then the sequence of corresponding minima, say $\{\boldsymbol{x}_n^*\}$, converges to a global minimizer $\boldsymbol{x}^*$ of the original constrained problem. This adaptation of the penalty parameters helps to balance the need to satisfy the constraints with the need to make progress toward the optimal solution.

2.6.2. Non-differentiable objective and constraints

We note that the approximation of the objective in Eq. (16) with Monte Carlo requires multiple runs of the expensive, forward, physics-based models involved, at each iteration of the optimization algorithm. In order to reduce the number of iterations required, especially when the dimension of the design space is higher, derivatives of the objective would be needed. In cases where the dimension of the design vector $\boldsymbol{x}$ is high, gradient-based methods are necessary. In turn, the computation of derivatives of $\mathcal{L}$ would necessitate derivatives of the outputs of the forward models with respect to the optimization variables $\boldsymbol{x}$. The latter are, however, unavailable due to the non-differentiability of the forward models. This is a common, restrictive feature of several physics-based simulators which in most cases of engineering practice are implemented in legacy codes that are run as black boxes. This lack of differentiability has been recognized as a significant roadblock by several researchers in recent years (Beaumont et al., 2002; Marjoram et al., 2003; Louppe et al., 2019; Cranmer et al., 2020; Shirobokov et al., 2020; Lucor et al., 2022; Oliveira et al., 2022; Agrawal and Koutsourelakis, 2023). In this work, we advocate Variational Optimization (Staines and Barber, 2013; Bird et al., 2018), which employs a differentiable bound on the non-differentiable objective. In the context of the current problem, we can write

$$\min_{\boldsymbol{x}} \int \left(\mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\right) p(\boldsymbol{b}|\boldsymbol{x},\boldsymbol{\varphi})\mathrm{d}\boldsymbol{b} \leq \underbrace{\int \left(\mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\right)p(\boldsymbol{b}|\boldsymbol{x},\boldsymbol{\varphi})q(\boldsymbol{x}|\boldsymbol{\theta})\ \mathrm{d}\boldsymbol{b}\ \mathrm{d}\boldsymbol{x}}_{U(\boldsymbol{\theta})}, \tag{17}$$

where $q(\boldsymbol{x}|\boldsymbol{\theta})$ is a density over the design variables $\boldsymbol{x}$ with parameters $\boldsymbol{\theta}$. If $\boldsymbol{x}^*$ yields the minimum of the objective $\mathbb{E}_{\boldsymbol{b}}[\mathcal{L}]$, then this can be achieved with a degenerate q that collapses to a Dirac-delta, that is, if $q(\boldsymbol{x}|\boldsymbol{\theta}) = \delta(\boldsymbol{x}-\boldsymbol{x}^*)$. For all other densities q or parameters $\boldsymbol{\theta}$, the inequality above would in general be strict. Hence, instead of minimizing $\mathbb{E}_{\boldsymbol{b}}[\mathcal{L}]$ with respect to $\boldsymbol{x}$, we can minimize the upper bound U with respect to $\boldsymbol{\theta}$. Under mild restrictions outlined by Staines and Barber (2012), the bound $U(\boldsymbol{\theta})$ is differential with respect to $\boldsymbol{\theta}$, and using the log-likelihood trick, its gradient can be rewritten as (Williams, 1992)

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{b}}\left[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x}|\boldsymbol{\theta})\,\mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\right] \\ &\approx \frac{1}{S}\sum_{i=1}^{S} \mathcal{L}(\boldsymbol{x}_i,\boldsymbol{b}_i,\boldsymbol{\lambda})\frac{\partial}{\partial\boldsymbol{\theta}}\log q(\boldsymbol{x}_i|\boldsymbol{\theta}). \end{aligned} \tag{18}$$

Both terms in the integrand can be readily evaluated which opens the door for a Monte Carlo approximation of the aforementioned expression by drawing samples $\boldsymbol{x}_i$ from $q(\boldsymbol{x}|\boldsymbol{\theta})$ and subsequently $\boldsymbol{b}_i$ from $p(\boldsymbol{b}|\boldsymbol{x}_i,\varphi^*)$.

2.6.3. Implementation considerations and challenges

While the Monte Carlo estimation of the gradient of the new objective $U(\boldsymbol{\theta})$ also requires running the expensive, forward models multiple times, it can be embarrassingly parallelized.

Obviously, convergence is impeded by the unavoidable Monte Carlo errors in the aforementioned estimates. In order to reduce them, we advocate the use of the baseline estimator proposed in Kool et al. (2019), which is based on the following expression:

$$\frac{\partial U}{\partial \theta} \approx \frac{1}{S-1}\sum_{i=1}^{S}\frac{\partial}{\partial \boldsymbol{\theta}}\log q(\boldsymbol{x}_i|\boldsymbol{\theta})\left(\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{b}_i,\boldsymbol{\lambda}) - \frac{1}{S}\sum_{j=1}^{S}\mathcal{L}\left(\boldsymbol{x}_j,\boldsymbol{b}_j,\boldsymbol{\lambda}\right)\right). \tag{19}$$

The estimator above is also unbiased as the one in Eq. (18); it does not imply any additional cost beyond the S samples and in addition exhibits lower variance as shown in Kool et al. (2019).

To efficiently compute the gradient estimators, we make use of the auto-differentiation capabilities of modern machine learning libraries. In the present study, `PyTorch` (Paszke et al., 2019) was utilized. For the stochastic gradient descent, the ADAM optimizer was used (Kingma and Ba, 2014). In the present study, $q(\boldsymbol{x}|\boldsymbol{\theta})$ was a Gaussian distribution with parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ representing mean and diagonal covariance, respectively. We say we have arrived at an optimal $\boldsymbol{x}^*$ when the q almost degenerates to a Dirac-delta, or colloquially, when the variance of q has converged and is considerably small. For completeness, the algorithm for the proposed optimization scheme is given in Algorithm 2. A schematic overview of the methods discussed in Sections 2.5 and 2.6 is presented in Figure 10.

Possible alternatives to the presented optimization scheme could be the popular gradient-free methods (Audet and Kokkolaras, 2016) like cBO and the extensions (Gardner et al., 2014) with appropriate adjustments to handle stochasticity, genetic algorithms (Banzhaf et al., 1998), COBYLA (Powell, 1994), and evolution strategies (Wierstra et al., 2014), to name a few. The gradient-free methods are known to perform poorly in high-dimensions (Moré and Wild, 2009). Since we use the approximate gradient information to move in the parametric space in the presented method, higher-dimensional optimization is feasible, as reported in Staines and Barber (2012) and Bird et al. (2018). The computational cost naturally scales exponentially with the number of design variables d, as the number of samples S per iteration is usually of the order $\mathcal{O}(d)$ (Salimans et al., 2017). Additionally, if

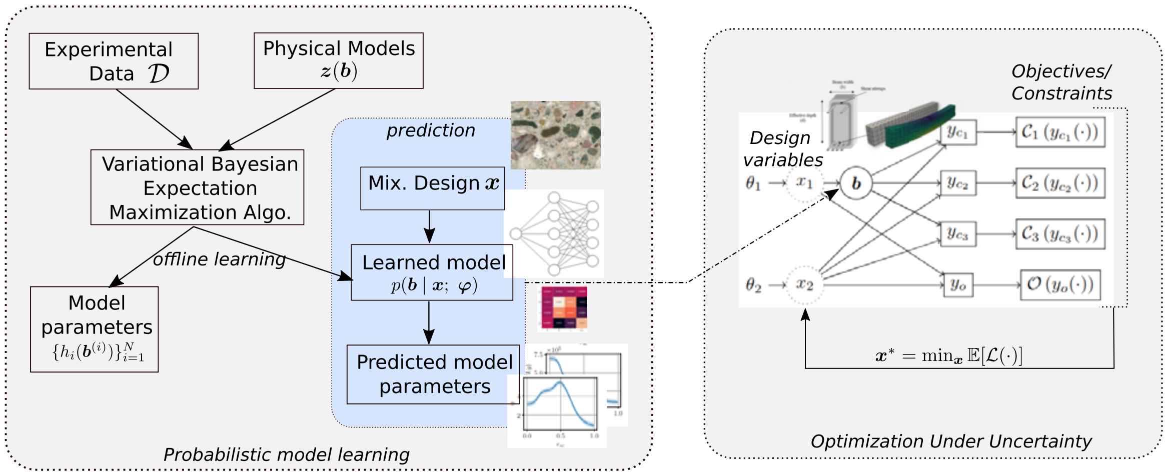

Experimental Data $\mathcal{D}$
Physical Models $z(\boldsymbol{b})$
Variational Bayesian Expectation Maximization Algo.
offline learning
Model parameters $\{h_i(\boldsymbol{b}^{(i)})\}_{i=1}^N$
prediction
Mix. Design $\boldsymbol{x}$
Learned model $p(\boldsymbol{b} \mid \boldsymbol{x};\ \varphi)$
Predicted model parameters
Probabilistic model learning
Objectives/ Constraints
Design variables
$\theta_1 \rightarrow x_1$
$\boldsymbol{b}$
$\theta_2 \rightarrow x_2$
y_{c_1}
y_{c_2}
y_{c_3}
y_o
$\mathcal{C}_1(y_{c_1}(\cdot))$
$\mathcal{C}_2(y_{c_2}(\cdot))$
$\mathcal{C}_3(y_{c_3}(\cdot))$
$\mathcal{O}(y_o(\cdot))$
$\boldsymbol{x}^* = \min_{\boldsymbol{x}} \mathbb{E}[\mathcal{L}(\cdot)]$
Optimization Under Uncertainty

Figure 10. *A schematic of the probabilistic model learning (left block) and the optimization under uncertainty (OUU; right block). The left block illustrates how the information from experimental data and physical models are fused together to learn the missing probabilistic link. This learned probabilistic link subsequently becomes a linchpin in predictive scenarios, particularly in downstream optimization tasks. The right block illustrates querying the learned probabilistic model to complete the missing link and interfacing the workflow describing the design variables, the physical models, and the key performance indicators. Subsequently, this integrated approach facilitates the execution of OUU as per the proposed methodology.*

a very expensive simulator is involved in the workflow, the core hours needed would naturally increase as generating each sample would become expensive. Fortunately, as the optimization scheme is embarrassingly parallelizable, physical time would be approximately equal to the time needed for one run of simulator times the number of optimization steps.

The presented optimization scheme has the following limitations, which we will attempt to address in the future. (1) Even after applying the baseline trick, one might observe some oscillations in $\boldsymbol{\theta}$ space near the optima. As suggested in Wierstra et al. (2014), these can potentially be eased by using natural gradients. (2) The proposed method can get trapped in local optima in highly multimodal surfaces. Casting the objective as $U(\boldsymbol{\theta})$ can be seen as a Gaussian blurred version of the original objective, the degree of smoothing being contingent upon the choice of initial . This smoothening essentially helps escape the local optima (also one of the strengths of the method). Still, for a highly multimodal surface (e.g., Ackley function), the smoothening might not be enough, leading to an early convergence to local optima. A safe bet would be to choose "large enough" , but this, in turn, might add to the computational cost (Wolpert and Macready, 1997). (3) The method might struggle for an objective surface with valleys (e.g., Rosenbrock function). If the dimension is not very high, a full matrix instead of the diagonal might help. (4) The applicability of the proposed workflow for multi-objective optimization is not explored in the present work, but since evolutionary strategies are a class of Variational Optimization and evolutionary strategies are known to work with multi-objective optimization problems (e.g., Deb et al., 2002), this might be an interesting research direction.

Algorithm 2 Black-box stochastic constrained optimization.

```
1: Input: distribution $q(\boldsymbol{x}|\boldsymbol{\theta})$ for the design variable $\boldsymbol{x}$, $n = 0$, learning rate $\eta$
2: $\boldsymbol{\theta}_0^0, \boldsymbol{\lambda}_1 \leftarrow$ choose starting point
3: for $k = 1, 2, \dots$ do
4:     while $\boldsymbol{\theta}_k$ not converged do
5:         Sample design variables and model parameters $\boldsymbol{x}_i \sim q(\boldsymbol{x}|\boldsymbol{\theta}_k^n)$, $\boldsymbol{b}_i \sim p(\boldsymbol{b}|\boldsymbol{x}_i, \boldsymbol{\varphi})$
6:         Farm the workflow with physics-based solvers for the samples in different machines and compute updated objective $\mathcal{L}(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}_k)$ for each of them    ▷ Eq. (14)
7:         Compute baseline
8:         Monte-Carlo gradient estimate $\nabla_{\boldsymbol{\theta}} U$    ▷ Eq. (19)
9:         $\boldsymbol{\theta}_k^{n+1} \leftarrow \boldsymbol{\theta}_k^n + \eta \nabla_{\boldsymbol{\theta}} U$    ▷ Stochastic Gradient Descent
10:        $n \leftarrow n + 1$
11:    end while
12:    if $\|\boldsymbol{\theta}_k^n - \boldsymbol{\theta}_{k-1}^n\| \leq \varepsilon$ then    ▷ Convergence condition
13:        break
14:    end if
15:    $\boldsymbol{\lambda}_{k+1} \leftarrow \boldsymbol{\lambda}_k$    ▷ Update penalty parameter
16:    $\boldsymbol{\theta}_{k+1}^0 \leftarrow \boldsymbol{\theta}_k^n$    ▷ Update parameters of distribution $q$
17: end for
18: return $\boldsymbol{\theta}_k^n$
```

3. Numerical experiments

This section presents the results of the data-based model learning (Section 3.1) and optimization (Section 3.2) methodological frameworks for the coupled workflow describing the concrete mix-design and structural performance as discussed in Figure 4.

3.1. Model-learning results

We report on the results obtained upon application of the method presented in Section 2.5 on the hydration and homogenization models (Table 4) along with the experimental data (Section 2.3).

Implementation details: We note that for the likelihood model (Eq. (3)) corresponding to the hydration model, the observed output $\hat{z}$ in the observed data $\mathcal{D} = \left\{ \hat{\boldsymbol{x}}^{(i)}, \hat{z}^{(i)} \right\}_{i=1}^{N}$ is the cumulative heat $\hat{\boldsymbol{Q}}$ and the covariance matrix $\Sigma_\ell = 3^2 \boldsymbol{I}_{\dim(\hat{\boldsymbol{Q}}) \times \dim(\hat{\boldsymbol{Q}})}$. The particular choice was made to account for the cumulative heat sensor noise of $\pm 4.5\,\mathrm{J/g}$ as reported in Gruyaert (2011). For the homogenization model, the $\hat{z} = [E_c, f_c]^T$ where E_c is the Young Modulus and f_c is the compressive strength of the concrete. The covariance matrix $\Sigma_\ell = \mathrm{diag}\left(4 \times 10^{18}, 2 \times 10^{12}\right) \mathrm{Pa}^2$. For both of the above, $\hat{\boldsymbol{x}}$ in the observed data $\mathcal{D}$ is the slag–binder mass ratio r_{sb}.

For both cases, a fully connected neural network is used to parameterize the mean of the conditional of model parameters $\boldsymbol{b}$ (Eq. (2)). The optimum number of hidden layers and nodes per layer was determined to be 1 and 30, respectively. The Tanh was chosen as the activation function for all layers. The L_2 weight regularization was employed to prevent over-fitting. We employed a learning rate of 10^{-2} for all the results reported here.

Owing to the intractability of the conditional posterior given in Eq. (6), we approximate it with MCMC, in particular we used the delayed rejection adaptive metropolis (DRAM) (Haario et al., 2006; Shahmoradi and Bagheri, 2020). The specific selection was motivated by two primary considerations. First, a gradient-free sampling strategy is imperative due to the absence of gradients in the physics-based models employed in this context. Second, we aim to introduce automation to the tuning of free parameters in the MCMC methods, ensuring a streamlined and efficient convergence process. In the DRAM sampler, we bound the target acceptance rate to be between 0.1 and 0.3.

Results: Figure 11 shows the learned probabilistic relation between the latent model parameters of the homogenization model and the slag–binder mass ratio r_{sb}. Out of the six available noisy datasets (Section 2.3), five were used for training and the dataset corresponding to $r_{\mathrm{sb}} = 0.5$ was used for testing. We access the predictive capabilities of the learned model by propagating the uncertainties forward via the homogenization model and analyzing the predictive density p_{pred} (Figure 10) as illustrated in Figure 12. We observe that the mechanical properties of concrete obtained by the homogenization model with learned probabilistic model predictions as the input envelops the ground truth.

Similarly, for the hydration model, Figure 13 shows the learned probabilistic relation between the latent model parameters $\left(B_1, B_2, \eta, Q_{\mathrm{pot}}\right)$ and the ratio r_{sb}. Out of the four available noisy datasets (Section 2.3) for $T = 20\,^\circ\mathrm{C}$, three were used for training and the dataset corresponding to $r_{\mathrm{sb}} = 0.5$ was used for testing. The value of E_a was taken from Gruyaert (2011). Figure 15 compares the experimental heat of hydration for different r_{sb} with the probabilistic predictions made using the learned

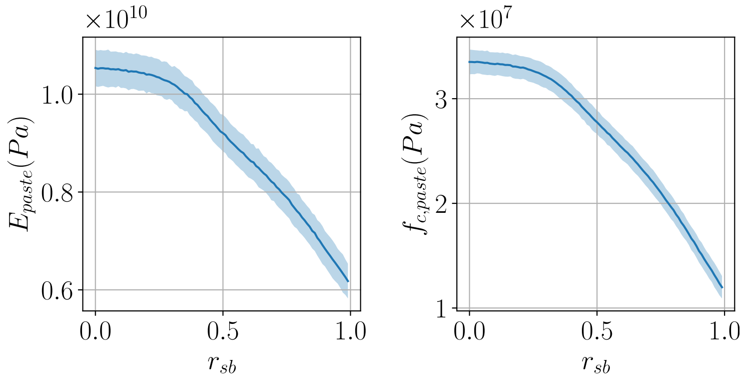


Figure 11. *Learned probabilistic relation between the homogenization model parameters and the slag–binder ratio r_{sb}. The solid line denotes the mean, and the shaded area denotes ± 2 times standard deviation.*

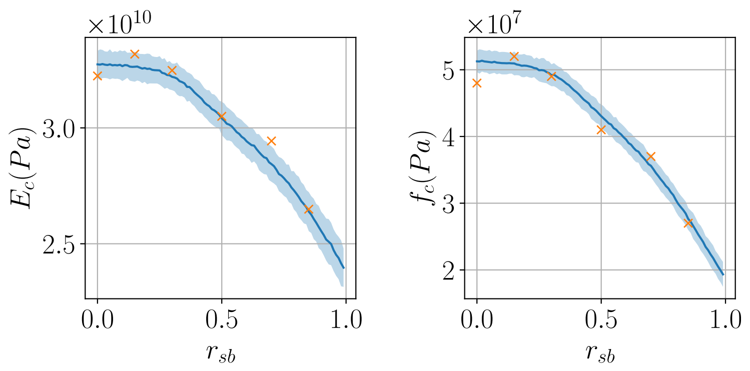


Figure 12. Predictive performance of the learned model corresponding to the homogenization process. The solid line is the predictive mean, and the shaded area is ± 2 times standard deviation. The crosses correspond to the noisy observed data.

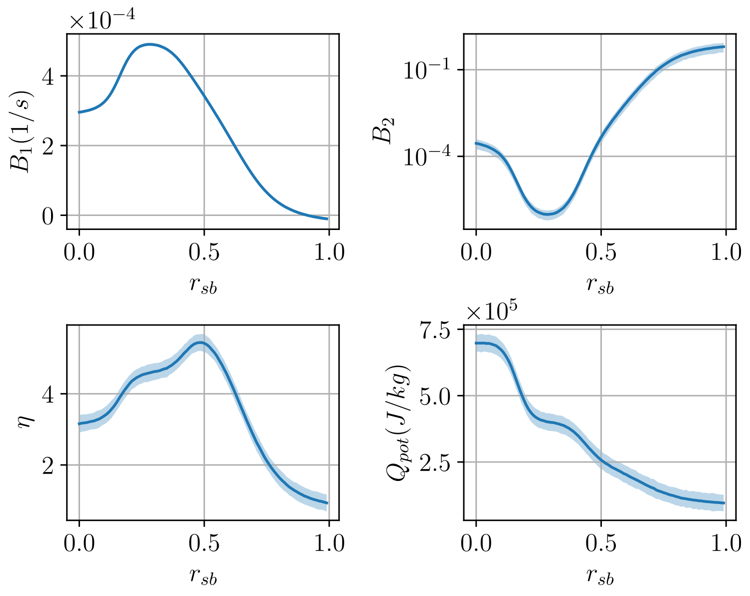
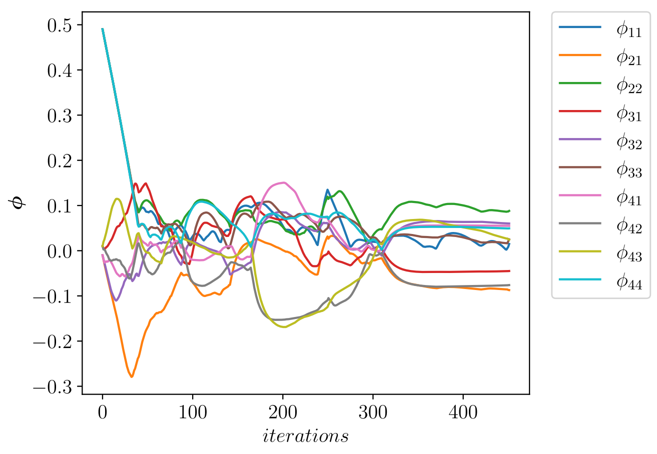
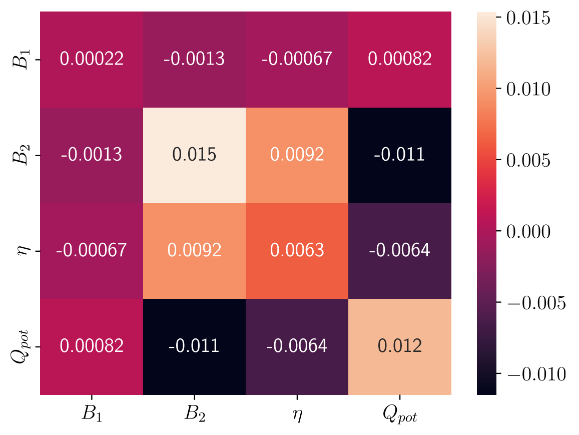


Figure 13. Learned probabilistic relation between the hydration model parameters and the slag–binder mass ratio r_{sb}. The solid line denotes the mean, and the shaded area denotes ± 2 times standard deviation.



(a)



	B_1	B_2	η	Q_{pot}
B_1	0.00022	-0.0013	-0.00067	0.00082
B_2	-0.0013	0.015	0.0092	-0.011
η	-0.00067	0.0092	0.0063	-0.0064
Q_{pot}	0.00082	-0.011	-0.0064	0.012

(b)

Figure 14. (a) Evolution of the entries ϕ_{ij} of the lower-triangular matrix $\mathbf{L}$ of the covariance matrix (Eq. (2)) with respect to EM iterations. (b) Heat map of the converged value of the covariance matrix $\mathbf{LL}^T$ of the probabilistic model corresponding to concrete hydration.

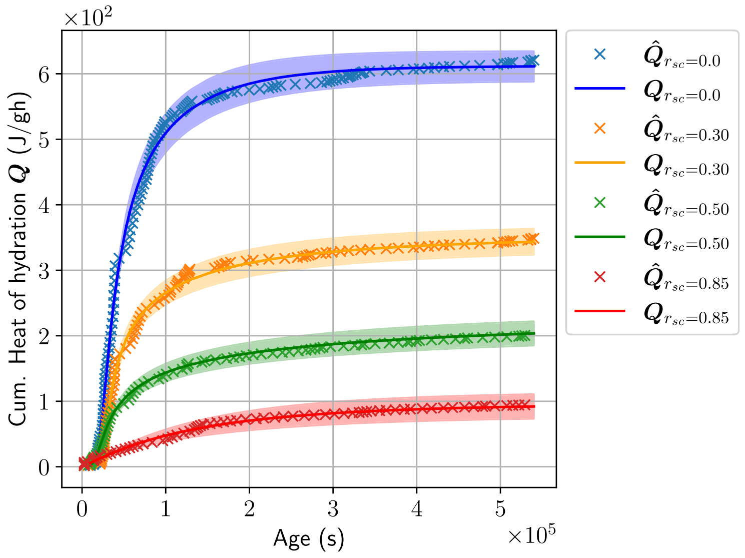


Figure 15. *Predictive performance of the learned model corresponding to the hydration process. The solid line is the predictive mean, and the shaded area is* ± 2 *times standard deviation. The crosses correspond to the noisy observed data.*

probabilistic model as an input to the hydration model. We observe that the predictions entirely envelop the ground truth data while accounting for the aleatoric noise present in the experimental data. Figure 14a shows the evolution of the entries of the covariance matrix of the conditional on the hydration model latent parameters $p(\boldsymbol{b}|\boldsymbol{x};\boldsymbol{\varphi})$. It serves as an indicator for the convergence of the VB-EM algorithm. The converged value of the covariance matrix is given by Figure 14b. It confirms the intricate correlation among the hydration model parameters, also reported in Figure B1. This is a general challenge with most physical models that are often overparameterized (at least for a given dataset) leading to multiple configurations of parameters with similar likelihood (see Figure 6).

At this point, it is crucial to (re)state that the training is performed using *indirect*, noisy data. It is encouraging to note that the learned models are able to account for the aleatoric uncertainty arising from the noise in the observed data and the epistemic uncertainty due to the finite amount of training data. The probabilistic model is able to learn relationships which were otherwise unavailable in literature, with the aid of physical models and (noisy) data. In this study, the training and validation of the model were somewhat constrained by the limited availability of data, a common challenge in engineering and physics applications. However, this limitation does not detract from the demonstration of our algorithmic framework. In future iterations of this work, an extensive set of experiments can be performed for a larger dataset.

3.2. Optimization results

With the learned probabilistic links as discussed in the previous section, we overcame the issue of forward and backward information flow bottleneck in the workflow connecting the design variables and KPIs relevant for constraints/objective (as discussed in Section 2.4). In this section, we report on the results obtained by performing OUU as discussed in Section 2.6 for the performance-based concrete design workflow. The design variables, objectives, and the constraints are detailed in Table 3. For the temperature constraint, we choose $T_{\text{limit}} = 60\,^\circ\text{C}$ and for the demolding time constraint, we choose 10 hours. To improve the numerical stability, we scale the variables, constraints, and objectives to make them of the order 1. To demonstrate the optimization scheme proposed, a simply supported beam is used as discussed in Section 1.1, with parameters given in Table D1. Colloquially, we aim to find the value(s) of slag–binder mass ratio r_{sb} and beam height h that minimize the objective, on average, while satisfying, on average, the aforementioned constraints.

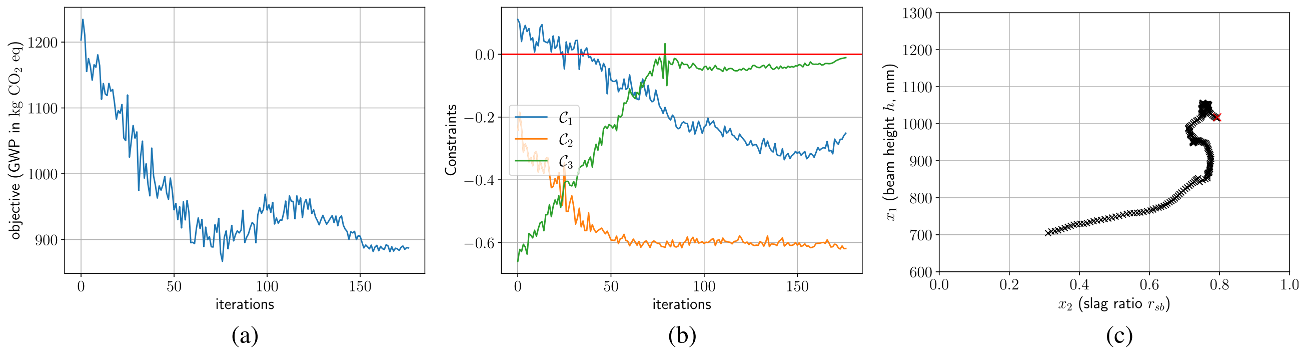


Figure 16. *(a) Evolution of the expected objective $\mathbb{E}_{\boldsymbol{b}}[\mathcal{O}]$ versus the number of iterations. The objective is the GWP of the beam. (b) Evolution of the expected constraints $\mathbb{E}_{\boldsymbol{b}}[\mathcal{C}_i]$ (which should all be negative) versus the number of iterations. $\mathcal{C}_1$ represents the beam design constraint, $\mathcal{C}_2$ represents the temperature constraint, and $\mathcal{C}_3$ gives the demolding time constraint. (c) Trajectory of the design variables (slag–binder mass ratio r_{sb} and the beam height h). The red cross represents the optimal value identified upon convergence.*

As discussed, the workflow for estimating the gradient is embarrassingly parallelizable. Hence, for each value considered in the design space, we call the ensemble of the workflows in parallel machines and collect the results. For the subsequent illustrations, a step size of 0.05 is utilized in the ADAM optimizer and $S = 100$ was the number of samples for gradient estimation. We set $\lambda_i = 1 \forall i \in \{1, \ldots, I\}$ as the starting value. Figure 16 shows the optimization results. In the design space, we start from values of design variables that violate the beam design constraints $\mathcal{C}_1$ as evident from Figure 16b. This activates the corresponding penalty term in the augmented objective (Eq. (14)), thus driving the design variables to satisfy the constraint (around iteration 40). Physically, this implies that the beam is not able to withstand the applied load for the given slag–binder ratio, beam height, and other material parameters (which are kept constant in the optimization procedure). As a result, the optimizer suggests to increase the beam height h in order to satisfy the constraint while also simultaneously increasing the slag–binder mass ratio r_{sb}, owing to the influence of the GWP objective (see Figure 16c). As it can be seen in Figure 16a, this leads to a reduction of the GWP because with the increase of the slag ratio, the Portland cement content, which is mainly responsible for the CO_2 emission, is ultimately reduced. In theory, the optimum value of the slag–binder mass ratio r_{sb} approaches one (meaning only slag in the mix) if only the GWP objective with no constraints were to be used in the optimization. But the demolding time constraint $\mathcal{C}_3$ penalizes the objective to limit the slag–binder r_{sb} ratio to be around 0.8 (see Figure 16c), since the evolution of mechanical properties is both much faster for Portland cement than for slag and at the same time the absolute values for strength and Young's modulus are higher. This s also evident in Figure 16b, when around iteration 80, the constraint violation line is crossed thus activating the penalty from $\mathcal{C}_3$. This also stops the nearly linear descent of the GWP objective. In the present illustrations, a value of 10 hours is chosen as the demolding time to demonstrate the procedure. In real-world settings, a manufacturer would be inclined to remove the formwork earlier so that it can be reused. But the lower the requirement of the demolding time, the higher the ratio of cement content required in the mix, leading to an increased hydration heat which in effect accelerates the reaction.

The oscillations in the objective and the constraints as seen in Figure 16a,b are due to the Monte Carlo noise in the gradient estimation. As per Eq. (17), the design variables are treated as random variables following a normal distribution. As discussed in Algorithm 2, the optimization procedure is assumed to converge when the standard deviations σ of the normal distribution attain small values (Figure 17), that is, when the normal degenerates to a Dirac-delta. The σ values stabilizing to relatively small values points toward the convergence of the algorithm.

The performance increase (in terms of GWP) is difficult to evaluate in the current setting. This is due to the fact that the constraint $\mathcal{C}_1$ is not fulfilled for the initial value of the design variables chosen for the optimization. It is to be highlighted that this is actually an advantage of the method—the user can start with

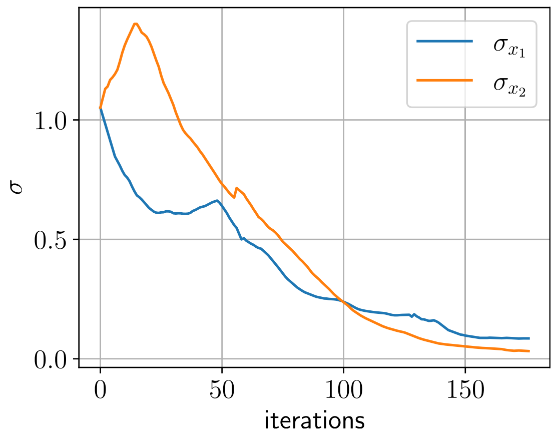


Figure 17. *Evolution of the standard deviations σ of the design variables to highlight the convergence of the optimization process. We note that the design variables are transformed and scaled in the optimization procedure.*

a reasonable design that still violates the constraints. In order to make a reasonable comparison, a design using only Portland cement (i.e., $r_{sb} = 0$) with only the load-bearing capacity as a constraint (beam design constraint $\mathcal{C}_1$) and the free parameter being the height of the beam was computed. This minimum height was found to be 77.5 cm with a corresponding GWP of the beam of 1,455 $kgCO_2eq$. Note that this design does not fulfill the temperature constraint $\mathcal{C}_2$ with a maximum temperature of 81°C. Another option for comparison is the first iteration number in the optimization procedure that fulfills all the constraints in expectation, which is the iteration number 30 with a GWP of 1,050 $kgCO_2eq$. In the subsequent iteration steps, this is further reduced to 900 $kgCO_2eq$ for the optimum value of the design variables obtained in the present study. This reduction in GWP is achieved by increasing the height of the beam to 100 cm while replacing Portland cement with BFS so that the mass fraction of slag–binder r_{sb} is 0.8. The addition of slag to the mixture decreases the strength of the material as illustrated in Figure 12, while at the same time, this decrease is compensated by an increased height. It is also informative to study the evolution of the (expected) constraints shown in Figure 16b. One observes that $\mathcal{C}_3$ (green line) associated with the demolding time is the most critical. Thus, in the current example, the GWP could be decreased even further when the time of demolding is extended (depending on the production process of removing the formwork).

4. Conclusion and outlook

We introduced a systematic design approach for the precast concrete industry in the pursuit of sustainable construction practices. It makes use of a holistic optimization framework which combines concrete mixture design with the structural simulation of the precast concrete element within an automated workflow. In this manner, various objectives and constraints, such as the environmental impact of the concrete element or its structural efficiency, can be considered.

The proposed holistic approach is demonstrated on a specific design problem, but should serve as a template that can be readily adapted to other design problems. The advocated black-box stochastic optimization procedure is able to deal with the challenges presented by general workflows, such as the presence of black-box models without derivatives, the effect of uncertainties, and nonlinear constraints. Furthermore, to complete the forward and backward information flow that is essential in the optimization procedure, a method to learn missing (probabilistic) links between the concrete mix design variables and model parameters from experimental data is presented. We note that, to the best of our knowledge, such a link is not available in the literature.

We demonstrated on the precast concrete element the integration of material and structural design in a joint workflow and showcased that this has the potential to decrease the objective, that is, the GWP. For the structural design, semi-analytical models based on the Eurocode are used, whereas the manufacturing process is simulated using a complex FE model. This illustrates the ability of the proposed procedure to combine multiple simulation tools of varying complexity, accounting for different parts of the life cycle. Hence, extending this in order to include, for example, additional load configurations, materials, or life cycle models, is straightforward. The present approach to treat the design process as a workflow, learning the missing links from data/models, and finally using this workflow in a global optimization is transferable to several other material, structural, and mechanical problems. Such extensions could readily include more complex design processes with an increased number of parameters and constraints (the latter due to multiple load configurations or limit states in a real structure). Furthermore, this procedure could be applied to problems involving a complete structure (e.g., bridge and building) instead of a single-element and potentially entailing advanced modeling features that include multiscale models to link material composition to material properties or improve the computation of the GWP using a complete life cycle analysis.

Data availability statement. The code and the data of this study are openly available at the project repository: https://github.com/BAMresearch/LebeDigital.

Acknowledgements. We acknowledge the support of David Alos Shepherd (Building Materials Technology, Karlsruhe Institute of Technology) for digitizing the experimental data. We thank Jonas Nitzler (TUM) for his advice on algorithmic implementations. Our sincere gratitude is extended to the reviewers, whose insightful feedback greatly enhanced the quality of our manuscript.

Author contribution. Conceptualization: P.-S.K., J.F.U.; Software: A.A., E.T.; Visualization: A.A., E.T.; Writing and reviewing code and paper: A.A., E.T., J.F.U.; Writing and reviewing paper: P.-S.K. All authors approved the final submitted draft.

Funding statement. The authors gratefully acknowledge the support of VDI Technologiezentrum GmbH and the Federal Ministry for Education and Research (BMBF) within the collaborative project “Lebenszyklus von Beton—Ontologieentwicklung für die Prozesskette der Betonherstellung.” A.A. and P.-S.K. received support through the subproject “Probabilistic machine learning for the calibration and validation of physics-based models of concrete” (FKZ-13XP5125B). E.T. and J.F.U. received support through the subproject “Ontologien und Workflows für Prozesse, Modelle und Abbildung in Datenbanken” (FKZ-13XP5125B).

Competing interest. The authors declare no competing interests.

Ethical standard. The research meets all ethical guidelines, including adherence to the legal requirements of the study country.

References

Acar E, Bayrak G, Jung Y, Lee I, Ramu P and Ravichandran SS (2021) Modeling, analysis, and optimization under uncertainties: A review. *Structural and Multidisciplinary Optimization 64*(5), 2909–2945.

ACI Committee 363 (2010) Report on high-strength concrete. Technical report ACI 363R-10, American Concrete Institute Committee 363, Farmington Hills, MI.

Agrawal A and Koutsourelakis PS (2023) A probabilistic, data-driven closure model for RANS simulations with aleatoric, model uncertainty. *Journal of Computational Physics 508*, 112982.

Agrawal A, Ravi K, Koutsourelakis P-S and Bungartz H-J (2023) Multi-fidelity constrained optimization for stochastic black-box simulators. *Workshop on Machine Learning and the Physical Sciences (NeurIPS 2023).*

Audet C and Kokkolaras M (2016) Blackbox and derivative-free optimization: theory, algorithms and applications. *Optimization and Engineering 17*, 1–2.

Baker N, Alexander F, Bremer T, Hagberg A, Kevrekidis Y, Najm H, Parashar M, Patra A, Sethian J, Wild S and Willcox K (2019) Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC.

Banzhaf W, Nordin P, Keller RE and Francone FD (1998) *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann Publishers Inc.

Beal MJ and Ghahramani Z (2003) The variational Bayesian EM algorithm for incomplete data: With application to scoring graphical model structures. *Bayesian Statistics 7*, 453–463.

Beaumont MA, Zhang W and Balding DJ (2002) Approximate Bayesian computation in population genetics. *Genetics 162*(4), 2025–2035.

Ben-Tal A and Nemirovski A (1999) Robust solutions of uncertain linear programs. *Operations Research Letters 25*(1), 1–13.

Bertsimas D, Brown DB and Caramanis C (2011) Theory and applications of robust optimization. *SIAM Review 53*(3), 464–501.

Bird T, Kunze J and Barber D (2018) Stochastic variational optimization. Preprint, arXiv:1809.04855 [cs, stat].

Bishop CM and Nasrabadi NM (2006) *Pattern Recognition and Machine Learning*, vol. 4. New York: Springer, p. 738.

Braga AM, **Silvestre JD and de Brito J** (2017) Compared environmental and economic impact from cradle to gate of concrete with natural and recycled coarse aggregates. *Journal of Cleaner Production 162*, 529–543.

Carette J and Staquet S (2016) Monitoring and modelling the early age and hardening behaviour of eco-concrete through continuous non-destructive measurements: Part II. Mechanical behaviour. *Cement and Concrete Composites 73*, 1–9.

Chakrabarty B (1992) Models for optimal design of reinforced concrete beams. *Computers and Structures 42*(3), 447–451.

Coello CC, **Hernández FS and Farrera FA** (1997) Optimal design of reinforced concrete beams using genetic algorithms. *Expert Systems with Applications 12*(1), 101–108.

Cranmer K, **Brehmer J and Louppe G** (2020) The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences 117*(48), 30055–30062.

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

DIN EN 1992-1-1 (2011) Eurocode 2: Bemessung und konstruktion von stahlbeton- und spannbetontragwerken—teil 1-1 allegmeine bemessungssregeln und regeln für den hochbau; deutsche fassung en 1992-1-1:2004 + ac:2010.

dos Santos NR, **Alves EC and Kripka M** (2023) Towards sustainable reinforced concrete beams: Multi-objective optimization for cost, CO_2 emission, and crack prevention. *Asian Journal of Civil Engineering 25*, 575–582.

Everard N and Tanner J (1966) *Reinforced Concrete Design*. Schaum's Outline Series. Schaum.

Fiacco AV and McCormick GP (1990) *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. SIAM.

Fleming N (2018) How artificial intelligence is changing drug discovery. *Nature 557*(7706), S55–S55.

Forsdyke JC, **Zviazhynski B**, **Lees JM and Conduit GJ** (2023) Probabilistic selection and design of concrete using machine learning. *Data-Centric Engineering 4*, e9.

Gardner JR, **Kusner MJ**, **Xu ZE**, **Weinberger KQ and Cunningham JP** (2014) Bayesian optimization with inequality constraints. *ICML 2014*, 937–945.

Gruyaert E (2011) Effect of blast-furnace slag as cement replacement on hydration, microstructure, strength and durability of concrete. PhD thesis, Ghent University.

Haario H, **Laine M**, **Mira A and Saksman E** (2006) DRAM: Efficient adaptive MCMC. *Statistics and Computing 16*, 339–354.

Haung EJ and Kirmser PG (1967) Minimum weight design of beams with inequality constraints on stress and deflection. *Journal of Applied Mechanics, Transactions of the ASME 34*, 999–1004.

Herlihy M, **Shavit N**, **Luchangco V and Spear M** (2020) *The art of multiprocessor programming*. Newnes.

Karniadakis GE, **Kevrekidis IG**, **Lu L**, **Perdikaris P**, **Wang S and Yang L** (2021) Physics-informed machine learning. *Nature Reviews Physics 3*(6), 422–440.

Karpatne A, **Kannan R and Kumar V** (2022) *Knowledge Guided Machine Learning: Accelerating Discovery Using Scientific Knowledge and Data*. CRC Press.

Kingma DP and Ba J (2014) Adam: A method for stochastic optimization. Preprint, arXiv:1412.6980.

Koller D and Friedman N (2009) *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Kondapally P, **Chepuri A**, **Elluri VP and Reddy BSK** (2022) Optimization of concrete mix design using genetic algorithms. *IOP Conference Series: Earth and Environmental Science 1086*(1), 012061.

Kool W, **van Hoof H and Welling M** (2019) Buy 4 reinforce samples, get a baseline for free! In *DeepRLStructPred@ICLR*. OpenReview.net.

Koutsourelakis P-S, **Zabaras N and Girolami M** (2016) Big data and predictive computational modeling. *Journal of Computational Physics 321*, 1252–1254.

Lisienkova L, **Shindina T**, **Orlova N and Komarova L** (2021) Optimization of the concrete composition mix at the design stage. *Civil Engineering Journal 7*(8), 1389–1405.

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.

Louppe G, **Hermans J and Cranmer K** (2019) Adversarial variational optimization of non-differentiable simulators. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1438–1447.

Lucor D, **Agrawal A and Sergent A** (2022) Simple computational strategies for more effective physics-informed neural networks modeling of turbulent natural convection. *Journal of Computational Physics 456*, 111022.

Marjoram P, **Molitor J**, **Plagnol V and Tavaré S** (2003) Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences 100*(26), 15324–15328.

Martins JR and Ning A (2021) *Engineering Design Optimization*. Cambridge: Cambridge University Press.

Menhorn F, **Augustin F**, **Bungartz H-J and Marzouk YM** (2017) A trust-region method for derivative-free nonlinear constrained stochastic optimization. Preprint, arXiv:1703.04156.

Miller SA, **Horvath A and Monteiro PJM** (2016) Readily implementable techniques can cut annual CO_2 emissions from the production of concrete by over 20%. *Environmental Research Letters 11*(7), 074029.

Mills RH (1966) Factors influencing cessation of hydration in water cured cement pastes. Highway Research Board Special Report.

Mohamed S, **Rosca M**, **Figurnov M and Mnih A** (2020) Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research 21*(1), 5183–5244.

Moré JJ and Wild SM (2009) Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization 20*(1), 172–191.

Mori T and Tanaka K (1973) Average stress in matrix and average elastic energy of materials with misfitting inclusions. *Acta Metallurgica 21*(5), 571–574.

Nežerka V, Hrbek V, Prošek Z, Somr M, Tesárek P and Fládr J (2018) Micromechanical characterization and modeling of cement pastes containing waste marble powder. *Journal of Cleaner Production 195*, 1081–1090.

Nežerka V and Zeman J (2012) A micromechanics-based model for stiffness and strength estimation of cocciopesto mortars. *Acta Polytechnica 52*(6), 28–37.

Nocedal J and Wright SJ (1999) *Numerical Optimization*. New York, NY: Springer.

Oliveira R, Scalzo R, Kohn R, Cripps S, Hardman K, Close J, Taghavi N and Lemckert C (2022) Bayesian optimization with informative parametric models via sequential monte carlo. *Data-Centric Engineering 3*, e5.

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J and Chintala S (2019) Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, 8024–8035.

Paya-Zaforteza I, Yepes V, Hospitaler A and González-Vidosa F (2009) CO_2-optimization of reinforced concrete frames by simulated annealing. *Engineering Structures 31*(7), 1501–1508.

Pierott R, Hammad AW, Haddad A, Garcia S and Falcón G (2021) A mathematical optimisation model for the design and detailing of reinforced concrete beams. *Engineering Structures 245*, 112861.

Powell MJ (1994) *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*. Dordrecht: Springer.

Qiu Z, Wu H, Elishakoff I and Liu D (2021) Data-based polyhedron model for optimization of engineering structures involving uncertainties. *Data-Centric Engineering 2*, e8.

Ruiz N, Schulter S and Chandraker M (2018) Learning to simulate. Preprint, arXiv:1810.02513.

Salimans T, Ho J, Chen X, Sidor S and Sutskever I (2017) Evolution strategies as a scalable alternative to reinforcement learning. Preprint, arXiv:1703.03864.

Shahmoradi A and Bagheri F (2020) ParaDRAM: A cross-language toolbox for parallel high-performance delayed-rejection adaptive metropolis Markov chain Monte Carlo simulations. Preprint, arXiv:2008.09589.

Shirobokov S, Belavin V, Kagan M, Ustyuzhanin A and Baydin AG (2020) Black-box optimization with local generative surrogates. *Advances in Neural Information Processing Systems 33*, 14650–14662.

Shobeiri V, Bennett B, Xie T and Visintin P (2023) Mix design optimization of concrete containing fly ash and slag for global warming potential and cost reduction. *Case Studies in Construction Materials 18*, e01832.

Snoek J, Larochelle H and Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems 25*, 2951–2959.

Staines J and Barber D (2012) Variational optimization. Preprint, arXiv:1212.4507 [cs, stat].

Staines J and Barber D (2013) *Optimization by variational bounding. In ESANN.*

Stránský J, Vorel J, Zeman J and Šejnoha M (2011) Mori–Tanaka-based estimates of effective thermal conductivity of various engineering materials. *Micromachines 2*(2), 129–149.

Wang I-J and Spall J (2003) Stochastic optimization with inequality constraints using simultaneous perturbations and penalty functions. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*. Maui, HI: IEEE, pp. 3808–3813.

Wierstra D, Schaul T, Glasmachers T, Sun Y, Peters J and Schmidhuber J (2014) Natural evolution strategies. *The Journal of Machine Learning Research 15*(1), 949–980.

Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning 8*(3), 229–256.

Wolpert D and Macready W (1997) No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation 1*(1), 67–82.

A. Homogenization

A.1. Approximation of elastic properties

The chosen method to homogenize the elastic, isotropic properties E and ν is the Mori–Tanaka homogenization scheme (Mori and Tanaka, 1973). It is a well-established, analytical homogenization method. The formulation uses bulk and shear moduli K and G. They are related to E and ν as $K = \frac{E}{3(1-2\nu)}$ and $G = \frac{E}{2(1+\nu)}$. The used Mori–Tanaka method assumes spherical inclusions in an infinite matrix and considers the interactions of multiple inclusions. The applied formulations follow the notation published in Nežerka and Zeman (2012) where this method is applied to successfully model the effective concrete stiffness for multiple types of inclusions. The general idea of this analytical homogenization procedure is to describe the overall stiffness of a body Ω, based on the properties of the individual phases, that is, the matrix and the inclusions. Each of the n phases is denoted by the index r, where $r = 0$ is defined as the matrix phase. The volume fraction of each phase is defined as

$$c^{(r)} = \frac{\|\Omega^{(r)}\|}{\|\Omega\|} \quad \text{for } r = 0, \ldots, n. \tag{A1}$$

The inclusions are assumed to be spheres, defined by their radius $R^{(r)}$. The elastic properties of each homogeneous and isotropic phase is given by the material stiffness matrix $\boldsymbol{L}^{(r)}$, here written in terms of the bulk and shear moduli K and G,

$$\boldsymbol{L}^{(r)} = 3K^{(r)}\boldsymbol{I}_{\mathrm{V}} + 2G^{(r)}\boldsymbol{I}_{\mathrm{D}} \quad \text{for } r = 0, \ldots, n, \tag{A2}$$

where $\boldsymbol{I}_{\mathrm{V}}$ and $\boldsymbol{I}_{\mathrm{D}}$ are the orthogonal projections of the volumetric and deviatoric components.

The method assumes that the micro-heterogeneous body Ω is subjected to a macroscale strain $\boldsymbol{\varepsilon}$. It is considered that for each phase a concentration factor $\boldsymbol{A}^{(r)}$ can be defined such that

$$\boldsymbol{\varepsilon}^{(r)} = \boldsymbol{A}^{(r)}\boldsymbol{\varepsilon} \quad \text{for } r = 0, \ldots, n, \tag{A3}$$

which computes the average strain $\boldsymbol{\varepsilon}^{(r)}$ within a phase, based on the overall strains. This can then be used to compute the effective stiffness matrix $\boldsymbol{L}_{\mathrm{eff}}$ as a volumetric sum over the constituents weighted by the corresponding concentration factor

$$\boldsymbol{L}_{\mathrm{eff}} = \sum_{r=0}^{n} c^{(r)}\boldsymbol{L}^{(r)}\boldsymbol{A}^{(r)} \quad \text{for } r = 0, \ldots, n. \tag{A4}$$

The concentration factors $\boldsymbol{A}^{(r)}$,

$$\boldsymbol{A}^{(0)} = \left(c^{(0)}\boldsymbol{I} + \sum_{r=1}^{n} c^{(r)}\boldsymbol{A}_{\mathrm{dil}}^{(r)} \right)^{-1}, \tag{A5}$$

$$\boldsymbol{A}^{(r)} = \boldsymbol{A}_{\mathrm{dil}}^{(r)}\boldsymbol{A}^{(0)} \quad \text{for } r = 1, \ldots, n, \tag{A6}$$

are based on the dilute concentration factors $\boldsymbol{A}_{\mathrm{dil}}^{(r)}$, which need to be obtained first. The dilute concentration factors are based on the assumption that each inclusion is subjected to the average strain in the matrix $\boldsymbol{\varepsilon}^{(0)}$; therefore,

$$\boldsymbol{\varepsilon}^{(r)} = \boldsymbol{A}_{\mathrm{dil}}^{(r)}\boldsymbol{\varepsilon}^{(0)} \quad \text{for } r = 1, \ldots, n. \tag{A7}$$

The dilute concentration factors neglect the interaction among phases and are only defined for the inclusion phases $r = 1, \ldots, n$. The applied formulation uses an additive volumetric–deviatoric split, where

$$\boldsymbol{A}_{\mathrm{dil}}^{(r)} = A_{\mathrm{dil,V}}^{(r)}\boldsymbol{I}_{\mathrm{V}} + A_{\mathrm{dil,D}}^{(r)}\boldsymbol{I}_{\mathrm{D}} \quad \text{for } r = 1, \ldots, n, \quad \text{with} \tag*{A8)}$$

$$A_{\mathrm{dil,V}}^{(r)} = \frac{K^{(0)}}{K^{(0)} + \alpha^{(0)}\left(K^{(r)} - K^{(0)}\right)}, \tag{A9}$$

$$A_{\mathrm{dil,D}}^{(r)} = \frac{G^{(0)}}{G^{(0)} + \beta^{(0)}\left(G^{(r)} - G^{(0)}\right)}. \tag{A10}$$

The auxiliary factors follow from the Eshelby solution as

$$\alpha^{(0)} = \frac{1 + \nu^{(0)}}{3\left(1 + \nu^{(0)}\right)} \quad \text{and} \quad \beta^{(0)} = \frac{2\left(4 - 5\nu^{(0)}\right)}{15\left(1 - \nu^{(0)}\right)}, \tag{A11}$$

where $\nu^{(0)}$ refers to the Poission ratio of the matrix phase. The effective bulk and shear moduli can be computed based on a sum over the phases

$$K_{\mathrm{eff}} = \frac{c^{(0)}K^{(0)} + \sum_{r=1}^{n} c^{(r)}K^{(r)}A_{\mathrm{dil,V}}^{(r)}}{c^{(0)} + \sum_{r=1}^{n} c^{(r)}A_{\mathrm{dil,V}}^{(r)}}, \tag{A12}$$

$$G_{\mathrm{eff}} = \frac{c^{(0)}G^{(0)} + \sum_{r=1}^{n} c^{(r)}G^{(r)}A_{\mathrm{dil,D}}^{(r)}}{c^{(0)} + \sum_{r=1}^{n} c^{(r)}A_{\mathrm{dil,D}}^{(r)}}. \tag{A13}$$

Based on the concept of Eq. (A3), with the formulations (Eqs. (A2), (A4), and (A5)), the average matrix stress is defined as

$$\boldsymbol{\sigma}^{(0)} = \boldsymbol{L}^{(0)}\boldsymbol{A}^{(0)}\boldsymbol{L}_{\mathrm{eff}}^{\;-1}\boldsymbol{\sigma}. \tag{A14}$$

A.1.1. Approximation of compressive strength

The estimation of the concrete compressive strength f_c follows the ideas of Nežerka et al. (2018). The procedure here is taken from the code provided in the link in Nežerka and Zeman (2012). The assumption is that a failure in the cement paste will cause the concrete to crack. The approach is based on two main assumptions. First, the Mori–Tanaka method is used to estimate the average stress within the matrix material $\boldsymbol{\sigma}^{(m)}$. The formulation is given in Eq. (A14). Second, the von Mises failure criterion of the average matrix stress is used to estimate the uniaxial compressive strength

$$f_c = \sqrt{3J_2}, \tag{A15}$$

with $J_2(\boldsymbol{\sigma}) = \frac{1}{2}\boldsymbol{\sigma}_\mathrm{D} : \boldsymbol{\sigma}_\mathrm{D}$ and $\boldsymbol{\sigma}_\mathrm{D} = \boldsymbol{\sigma} - \frac{1}{3}\mathrm{trace}(\boldsymbol{\sigma})\boldsymbol{I}$. It is achieved by finding a uniaxial macroscopic stress $\boldsymbol{\sigma} = \begin{bmatrix} -f_{c,\mathrm{eff}} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\mathrm{T}$, which exactly fulfills the von Mises failure criterion (Eq. (A15)) for the average stress within the matrix $\boldsymbol{\sigma}^{(m)}$. The procedure here is taken from the code provided in the link in Nežerka and Zeman (2012). First, a J_2^test is computed for a uniaxial test stress $\boldsymbol{\sigma}^\mathrm{test} = \begin{bmatrix} f^\mathrm{test} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\mathrm{T}$. Then the matrix stress $\boldsymbol{\sigma}^{(m)}$ is computed based on the test stress following Eq. (A14). This is used to compute the second deviatoric stress invariant $J_2^{(m)}$ for the average matrix stress. Finally, the effective compressive strength is estimated as

$$f_{c,\mathit{eff}} = \frac{J_2^\mathrm{test}}{J_2^{(m)}} f^\mathrm{test}. \tag{A16}$$

A.1.2. Approximation of thermal conductivity

Homogenization of the thermal conductivity is based on the Mori–Tanaka method as well. The formulation is similar to Eqs. (A12) and (A13). The expressions are taken from Stránský et al. (2011). The thermal conductivity χ_eff is computed as

$$\chi_\mathrm{eff} = \frac{c^{(m)}\chi^{(m)} + c^{(i)}\chi^{(i)}A_\chi^{(i)}}{c^{(m)} + c^{(i)}A_\chi^{(i)}} \quad \text{and} \tag{A17}$$

$$A_\chi^{(i)} = \frac{3\chi^{(m)}}{2\chi^{(m)} + \chi^{(i)}}. \tag{A18}$$

B. FE concrete model

B.1. Modeling of the temperature field

The temperature distribution is generally described by the heat equation as

$$\rho C \frac{\partial T}{\partial t} = \nabla \cdot (\lambda \nabla T) + \frac{\partial Q}{\partial t} \tag{B1}$$

with λ the effective thermal conductivity, C the specific heat capacity, ρ the density, and ρC the volumetric heat capacity. The volumetric heat Q due to hydration is also called the latent heat of hydration, or the heat source. In this article, the density, the thermal conductivity, and the volumetric heat capacity to be constants are assumed to be sufficiently accurate for our purpose, even though there are more elaborate models taking into account the effects of temperature, moisture, and/or the hydration.

B.1.1. Degree of hydration α

The DOH α is defined as the ratio between the cumulative heat Q at time t and the total theoretical volumetric heat by complete hydration Q_∞:

$$\alpha(t) = \frac{Q(t)}{Q_\infty}, \tag{B2}$$

assuming a linear relation between the DOH and the heat development. Therefore, the time derivative of the heat source $\dot{Q}$ can be rewritten in terms of α,

$$\frac{\partial Q}{\partial t} = \frac{\partial \alpha}{\partial t} Q_\infty. \tag{B3}$$

Approximated values for the total potential heat range between 300 and 600 J/g for binders of different cement types, for example, OPC $Q_\infty = 375 - 525\,\mathrm{J/g}$ or Pozzolanic cement $Q_\infty = 315 - 420$ J/g.

B.1.2. Affinity

The heat release can be modeled based on the chemical affinity A of the binder. The hydration kinetics are defined as a function of affinity at a reference temperature $\tilde{A}$ and a temperature dependent scale factor a

$$\dot{\alpha} = \tilde{A}(\alpha)a(T). \tag{B4}$$

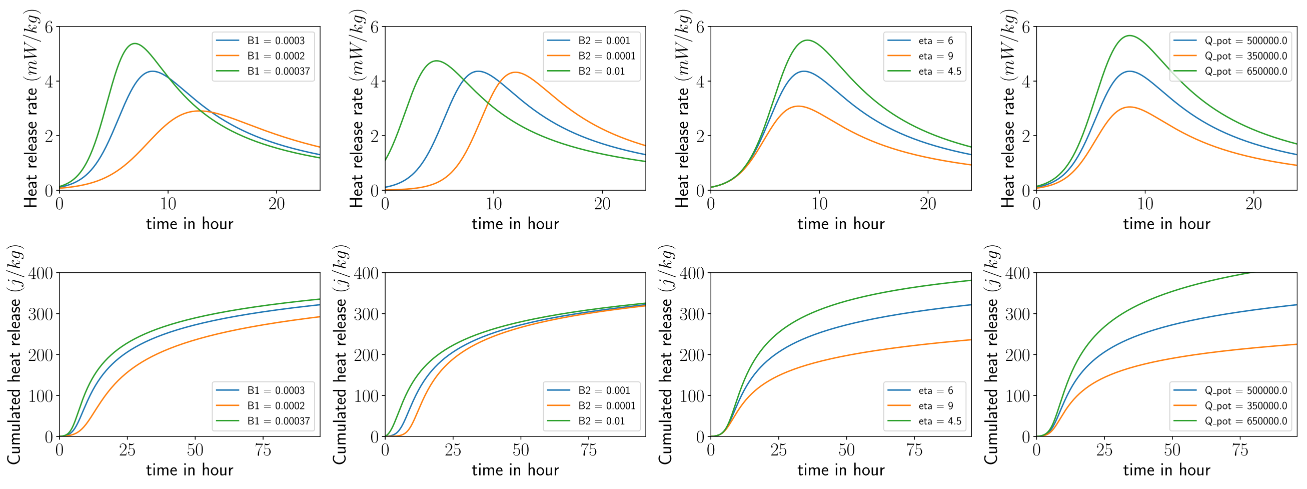


Figure B1. *Influence of the hydration parameters on the heat release rate and the cumulative heat release.*

The reference affinity, based on the DOH, is approximated by

$$\tilde{A}(\alpha) = B_1\left(\frac{B_2}{\alpha_{\max}} + \alpha\right)(\alpha_{\max} - \alpha)\exp\left(-\eta\frac{\alpha}{\alpha_{\max}}\right), \tag{B5}$$

where B_1 and B_2 are coefficients depending on the binder. The scale function is given as

$$a = \exp\left(-\frac{E_a}{R}\left(\frac{1}{T} - \frac{1}{T_{\mathrm{ref}}}\right)\right). \tag{B6}$$

An example function to approximate the maximum DOH based on the water to cement mass ratio r_{wc}, by Mills (1966), is the following:

$$\alpha_{\max} = \frac{1.031 r_{\mathrm{wc}}}{0.194 + r_{\mathrm{wc}}}. \tag{B7}$$

This refers to Portland cement. Figure B1 shows the influence of the three numerical parameters B_1, B_2, η and the potential heat release Q_∞ on the heat release rate as well as on the cumulative heat release.

B.1.3. Discretization and solution

Using Eq. (B3) in Eq. (B1), the heat equation is given as

$$\rho C\frac{\partial T}{\partial t} = \nabla\cdot(\lambda\nabla T) + Q_\infty\frac{\partial\alpha}{\partial t}. \tag{B8}$$

Now we apply a backward Euler scheme

$$\dot{T} = \frac{T^{n+1} - T^n}{\Delta t} \quad \text{and} \tag{B9}$$

$$\dot{\alpha} = \frac{\Delta\alpha}{\Delta t} \quad \text{with} \quad \Delta\alpha = \alpha^{n+1} - \alpha^n \tag{B10}$$

and drop the index $n+1$ for readability to obtain

$$\rho CT - \Delta t\nabla\cdot(\lambda\nabla T) - Q_\infty\Delta\alpha = \rho CT^n. \tag{B11}$$

Using Eqs. (B10) and (B4), a formulation for $\Delta\alpha$ is obtained:

$$\Delta\alpha = \Delta t\tilde{A}(\alpha)a(T). \tag{B12}$$

We define the affinity in terms of α^n and $\Delta\alpha$ to solve for $\Delta\alpha$ on the quadrature point level

$$\tilde{A} = B_1\exp\left(-\eta\frac{\Delta\alpha + \alpha^n}{\alpha_{\max}}\right)\left(\frac{B_2}{\alpha_{\max}} + \Delta\alpha + \alpha^n\right)\cdot(\alpha_{\max} - \Delta\alpha - \alpha^n). \tag{B13}$$

Now we can solve the nonlinear function

$$f(\Delta\alpha) = \Delta\alpha - \Delta t\tilde{A}(\Delta\alpha)a(T) = 0 \tag{B14}$$

using an iterative Newton–Raphson solver.

B.2. Coupling material properties to degree of hydration

B.2.1. Compressive strength

The compressive strength in terms of the DOH can be approximated using an exponential function (cf. Carette and Staquet, 2016):

$$f_c(\alpha) = \alpha(t)^{a_{f_c}} f_{c\infty}. \tag{B15}$$

This model has two parameters, $f_{c\infty}$, the compressive strength of the parameter at full hydration, $\alpha = 1$, and a_{f_c} the exponent, which is a material parameter that characterizes the temporal evolution.

The first parameter could theoretically be obtained through experiments. However, the total hydration can take years. Therefore, we can compute it using the 28 days values of the compressive strength and the corresponding DOH:

$$f_{c\infty} = \frac{f_{c28}}{\alpha_{28}{}^{a_{f_c}}}. \tag{B16}$$

B.2.2. Young's modulus

The publication (Carette and Staquet, 2016) proposes a model for the evolution of the Young modulus assuming an initial linear increase of the Young modulus up to a DOH α_t:

$$E(\alpha) = \begin{cases} E_\infty \dfrac{\alpha(t)}{\alpha_t} \alpha_t{}^{a_E} & \text{for } \alpha < \alpha_t \\ E_\infty \alpha(t)^{a_E} & \text{for } \alpha \geq \alpha_t. \end{cases} \tag{B17}$$

Contrary to other publications, no dormant period is assumed. Similarly to the strength standardized testing of the Young modulus is done after 28 days, E_{28}. To effectively use these experimental values, E_∞ is approximated as

$$E_\infty = \frac{E_{28}}{\alpha_{28}{}^{a_E}}, \tag{B18}$$

using the approximated DOH.

B.3. Constraints

The FEM simulation is used to compute two practical constraints relevant to the precast concrete industry. At each time step, the worst point is chosen to represent the part, therefore ensuring that the criterion is fulfilled in the whole domain. The first constraint limits the maximum allowed temperature. The constraint is computed as the normalized difference between the maximum temperature reached T_{max} and the temperature limit T_{limit}

$$\mathcal{C}_{\text{T}} = \frac{T_{\text{max}} - T_{\text{limit}}}{T_{\text{limit}}}, \tag{B19}$$

where $\mathcal{C}_{\text{T}} > 0$ is not admissible, as the temperature limit 60°C has been exceeded.

The second constraint is the estimated time of demolding. This is critical, as the manufacturer has a limited number of forms. The faster the part can be demolded, the faster it can be reused, increasing the output capacity. On the other hand, the part must not be demolded too early, as it might get damaged while being moved. To approximate the minimal time of demolding, a constraint is formulated based on the local stresses $\mathcal{C}_\sigma$. It evaluates the Rankine criterion for the principal tensile stresses, using the yield strength of steel f_{yk} and a simplified Drucker–Prager criterion, based on the evolving compressive strength of the concrete f_c,

$$\mathcal{C}_\sigma = \max \begin{cases} \mathcal{C}_{\text{RK}} = \dfrac{\|\sigma_t'\| - f_{\text{yk}}}{f_{\text{yk}}} \\ \mathcal{C}_{\text{DP}} = \dfrac{\sqrt{\frac{1}{2} I_1^2 - I_2} - \frac{f_c^3}{\sqrt{3}}}{f_c} \end{cases}, \tag{B20}$$

where $\mathcal{C}_\sigma > 0$ is not admissible. In contrast to standard yield surfaces, the value is normalized, to be unit less. This constraint aims to approximate the compressive failure often simulated with plasticity and the tensile effect of reinforcement steel. As boundary conditions, a simply supported beam under its own weight has been chosen to approximate possible loading conditions while the part is moved. This constraint is evaluated for each time step in the simulation. The critical point in time is approximated where $\mathcal{C}_\sigma(t_{\text{crit}}) = 0$. This is normalized with the prescribed time of demolding to obtain a dimensionless constraint.

C. Beam design

We follow the design code (DIN EN 1992-1-1, 2011) for a singly reinforced beam, which is a reinforced concrete beam with reinforcement only at the bottom. The assumed cross-section is rectangular.

C.1. *Maximum bending moment*

Assuming a simply supported beam with a given length l in millimeters, a distributed load q in Newton per millimeters and a point load F in Newton per millimeters, the maximum bending moment $M_{\max}$ in Newton per square millimeter is computed as

$$M_{\max} = q\frac{l^2}{8} + F\frac{l}{4}. \tag{C1}$$

The applied loads already incorporate any required safety factors.

C.2. *Computing the minimal required steel reinforcement*

Given a beam with the height h in millimeters, a concrete cover of c in millimeters, a steel reinforcement diameter of d in millimeters for the longitudinal bars, and a bar diameter of d in millimeters for the transversal reinforcement also called stirrups,

$$h_{\text{eff}} = h - c - d_{\text{st}} - \frac{1}{2}d. \tag{C2}$$

According to the German norm standard safety factors are applied, $\alpha_{\text{cc}} = 0.85$, $\gamma_c = 1.5$, and $\gamma_s = 1.15$, leading to the design compressive strength for concrete f_{cd} and the design tensile yield strength f_{ywd} for steel

$$f_{\text{cd}} = \alpha_{\text{cc}}\frac{f_{\text{c}}}{\gamma_{\text{c}}}, \tag{C3}$$

$$f_{\text{ywd}} = \frac{f_{\text{yk}}}{\gamma_{\text{s}}}, \tag{C4}$$

where f_c denotes the concrete compressive strength and f_{yk} the steel's tensile yield strength.

To compute the force applied in the compression zone, the lever arm of the applied moment z is given by

$$z = h_{\text{eff}}\left(0.5 + \sqrt{0.25 - 0.5\mu}\right), \quad \text{with} \tag{C5}$$

$$\mu = \frac{M_{\max}}{bh_{\text{eff}}^2 f_{\text{cd}}}. \tag{C6}$$

The minimum required steel $A_{s,\text{req}}$ is then computed based on the lever arm, the design yield strength of steel and the maximum bending moment, as

$$A_{s,\text{req}} = \frac{M_{\max}}{f_{\text{ywd}}z}. \tag{C7}$$

C.3. *Optimization constraints*

C.3.1. *Compressive strength constraint*

Based on Eq. (C6), we define the compressive strength constraint as

$$\mathcal{C}_{\text{fc}} = \mu - 0.5, \tag{C8}$$

where $\mathcal{C}_{\text{fc}} > 0$ is not admissible, as there is no solution for Eq. (C6).

C.3.2. *Geometrical constraint*

The geometrical constraint checks that the required steel area $A_{s,\text{req}}$ does not exceed the maximum steel area $A_{s,\max}$ that fits inside the design space. For our example, we assume the steel reinforcement is only arranged in a single layer. This limits the available space for rebars in two ways, by the required minimal spacing $s_{\min}$ between the bars, to allow concrete to pass, and by the required space on the outside, the concrete cover c, and stirrups diameter d_{st}. To compute $A_{s,\max}$, the maximum number for steel bars $n_{s,\max}$ and the maximum diameter $d_{\max}$ from a given list of admissible diameters are determined that fulfill

$$s \geq s_{\min}, \quad \text{with} \tag{C9}$$

$$s = \frac{b - 2c - 2d_{\text{st}} - n_{s,\max}d_{\max}}{n_{s,\max} - 1} \quad \text{and} \tag{C10}$$

$$n_{s,\max} \geq 2. \tag{C11}$$

According to DIN1992-1-1 (2011), the minimum spacing between two bars $s_{\min}$ is given by the minimum of the concrete cover (2.5 cm) and the rebar diameter. The maximum possible reinforcement is given by

$$A_{s,\max} = n_s \pi \left(\frac{d}{2}\right)^2. \tag{C12}$$

The geometry constraint is computed as

$$\mathcal{C}_g = \frac{A_{s,\mathrm{req}} - A_{s,\max}}{A_{s,\max}}, \tag{C13}$$

where $\mathcal{C}_g > 0$ is not admissible, as the required steel area exceeds the available space.

C.3.3. Combined beam constraint

To simplify the optimization procedure, the two constraints are combined into a single one by using the maximum value:

$$\mathcal{C}_{\mathrm{beam}} = \max\left(\mathcal{C}_g, \mathcal{C}_{\mathrm{fc}}\right). \tag{C14}$$

Evidently, this constraint is also defined as: $\mathcal{C}_{\mathrm{beam}} > 0$ is not admissible.

D. Parameter tables

This is the collection of the used parameters for the various example calculations.

Table D1. *Parameters of the simply supported beam for the computation of the steel reinforcement*

Name	Value	Unit
Length	1,000	cm
Width	350	mm
Height	450	mm
Steel yield strength	300	$\mathrm{N/mm^2}$
Diameter stirrups	10	mm
Minimal concrete cover	2.5	cm
Load	50	kN
Concrete compressive strength	40	$\mathrm{N/mm^2}$

D
Multi-fidelity Constrained Optimization for Stochastic Black-Box Simulators

Multi-fidelity Constrained Optimization for Stochastic Black-Box Simulators

Atul Agrawal[*], **Phaedon-Stelios Koutsourelakis**
Professorship of Data-Driven Materials Modelling
Technical University of Munich
Munich, Germany
{atul.agrawal,p.s.koutsourelakis}@tum.de

Kislaya Ravi[*], **Hans-Joachim Bungartz**
Chair of Scientific Computing
Technical University of Munich, Germany
{kislaya.ravi,bungartz}@tum.de

Abstract

Constrained optimization of the parameters of a simulator plays a crucial role in a design process. These problems become challenging when the simulator is stochastic, computationally expensive, and the parameter space is high-dimensional. One can efficiently perform optimization only by utilizing the gradient with respect to the parameters, but these gradients are unavailable in many legacy, black-box codes. We introduce the algorithm Scout-Nd (Stochastic Constrained Optimization for N dimensions) to tackle the issues mentioned earlier by efficiently estimating the gradient, reducing the noise of the gradient estimator, and applying multi-fidelity schemes to further reduce computational effort. We validate our approach on standard benchmarks, demonstrating its effectiveness in optimizing parameters highlighting better performance compared to existing methods.

1 Introduction

Physics-based simulators are used across fields of engineering and science to drive research [1], and more recently used to generate synthetic training data for machine learning related tasks [2]. A common challenge is finding optimum parameters given some objective subject to some constraints. High-dimensional parameter space and stochastic objective function make the optimization non-trivial.

Gradient-based methods have been shown to work well when the derivative is available [3, 4, 5, 6]. However, for optimization/inference tasks involving physics-based simulators, only black-box evaluations of the objective are often possible(e.g., legacy solvers). It is commonly called simulation based inference(SBI)/optimization [1, 7]. In such cases, one resort to gradient-free optimization [8], for example, genetic algorithms [9] or Bayesian Optimization and their extensions [10, 11]. The gradient-free methods perform poorly on high-dimensional parametric spaces [8]. More recently, stochastic gradient estimators [12] have been used to estimate gradients of black-box functions and, hence, perform gradient-based optimization [13, 14, 15, 16]. However, they do not account for the constraints.

This work introduces a novel approach for constrained stochastic optimization involving stochastic black-box simulators with high-dimensional parametric dependency. We draw inspiration from [17,

[*]equal contribution

18] to estimate the gradients, extended it to include constraints and employed multi-fidelity strategies to limit the number of function calls, as the cost of running the simulator can be high. We choose popular gradient-free constrained optimization methods like Constrained Bayesian Optimization (cBO)[19] and COBYLA [20] to compare our method on standard benchmark problems.

2 Methodology

Problem statement We are given a scalar valued function $f(\boldsymbol{x}, \boldsymbol{b})$ and a set of constraints $\mathcal{C}(\boldsymbol{x}, \boldsymbol{b}) = \{\mathcal{C}_1(\boldsymbol{x}, \boldsymbol{b}), \ldots, \mathcal{C}_I(\boldsymbol{x}, \boldsymbol{b})\}$, where $\boldsymbol{x} \in \mathbb{R}^d$ are the deterministic parameters and $\boldsymbol{b}$ represents a random vector [13]. The uncertainty may be caused by a lack of knowledge about the parameters or the inherent noise in the system. The objective f or the constraints $\mathcal{C}$ depend implicitly on the output of the black-box simulator. Our task is to minimize the function $f(\boldsymbol{x}, \boldsymbol{b})$ with respect to $\boldsymbol{x}$ subject to the constraints $\mathcal{C}(\boldsymbol{x}, \boldsymbol{b})$. Because of the stochastic nature of the problem, we will optimize the objective function with respect to a robustness measure [21, 22]. In this work, we will only consider the expectation as a robustness measure, in which case, the problem can be stated as follows:

$$\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}}[f(\boldsymbol{x}, \boldsymbol{b})], \quad \text{s.t} \;\; \mathbb{E}_{\boldsymbol{b}}[\mathcal{C}_i(\boldsymbol{x}, \boldsymbol{b})] \leq 0, \;\; \forall i \in \{1, \ldots, I\} \tag{1}$$

In addition to that, the gradient of the objective function and the constraint is unavailable. Hence, one cannot directly apply gradient-based optimization methods.

2.1 Constraint augmentation

We cast the constrained optimization problem (Eq. (1)) to an unconstrained one using penalty-based methods [23, 24]. We define an augmented objective function $\mathcal{L}$ as follows:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) = f(\boldsymbol{x}, \boldsymbol{b}) + \sum_{i=i}^{I} \lambda_i \max\left(\mathcal{C}_i(\boldsymbol{x}, \boldsymbol{b}), 0\right) \tag{2}$$

where $\lambda_i > 0$ is the penalty parameter for the i^{th} constraint and the $\max(\cdot, \cdot)$ controls the magnitude of penalty applied. Incorporating the augmented objective (Eq. (2)) in the Eq. (1), one can arrive at the following optimization problem:

$$\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}}[\mathcal{L}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda})] \tag{3}$$

The expectation is approximated by Monte Carlo which induces noise and necessitates stochastic optimization methods. We alleviate the dependence on the penalty parameter $\boldsymbol{\lambda}$ by using the sequential unconstrained minimization technique (SUMT) algorithm [25] where one starts with a small penalty term and gradually increases its value.

2.2 Estimation of derivative

The direct computation of derivatives of $\mathcal{L}$ with respect to the optimization variables $\boldsymbol{x}$ is not feasible because of the unavailability of the gradients of the objective function and the constraints. One notes many active research threads across disciplines which are trying to tackle this bottleneck [1, 14, 26, 27, 4, 15, 15]. We draw inspiration from the Variational Optimization [17, 28, 16], which constructs an upper bound of the objective function as shown below:

$$\min \int \mathcal{L}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) p(\boldsymbol{b}) d\boldsymbol{b} \leq \int \mathcal{L}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) p(\boldsymbol{b}) q(\boldsymbol{x} \mid \boldsymbol{\theta}) d\boldsymbol{b} d\boldsymbol{x} = U(\boldsymbol{\theta}) \tag{4}$$

where $q(\boldsymbol{x} \mid \boldsymbol{\theta})$ is a density over the design variables $\boldsymbol{x}$ with parameters $\boldsymbol{\theta}$. If $\boldsymbol{x}^*$ yields the minimum of the objective $\mathbb{E}_{\boldsymbol{b}}[\mathcal{L}]$, then this can be achieved with a degenerate q that collapses to a Dirac-delta, i.e. if $q(\boldsymbol{x} \mid \boldsymbol{\theta}) = \delta(\boldsymbol{x} - \boldsymbol{x}^*)$. The inequality above would generally be strict for all other densities q or parameters $\boldsymbol{\theta}$. Hence, instead of minimizing $\mathbb{E}_{\boldsymbol{b}}[\mathcal{L}]$ with respect to $\boldsymbol{x}$, we can minimize the upper bound $U(\boldsymbol{\theta})$ with respect to the distribution parameters $\boldsymbol{\theta}$. Under mild restrictions outlined by [18], the bound $U(\boldsymbol{\theta})$ is differential w.r.t $\boldsymbol{\theta}$. One can evaluate the gradient of $U(\boldsymbol{\theta})$ as shown below:

$$\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}, \boldsymbol{b}}\left[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta}) \mathcal{L}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda})\right] \tag{5}$$

The Monte Carlo estimation of the expectation shown in Eq. (5) is as follows:

$$\frac{\partial U}{\partial \theta} \approx \frac{1}{S} \sum_{i=1}^{S} \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}) \frac{\partial}{\partial \boldsymbol{\theta}} \log q\left(\boldsymbol{x}_i \mid \theta\right) \tag{6}$$

The Eq. (6) is known as the score function estimator [29], which also appears in the context of reinforcement learning [30]. The gradient estimation can be computationally expensive as each step will involve calling the simulator S number of times. This step is can be easily parallelized.

2.3 Variance reduction

To reduce the mean square error of the estimator in Eq. (6), we propose the use of baseline discussed in [31] as shown below:

$$\frac{\partial U}{\partial \theta} \approx \frac{1}{S} \sum_{i=1}^{S} \frac{\partial}{\partial \boldsymbol{\theta}} \log q\left(\boldsymbol{x}_i \mid \boldsymbol{\theta}\right) \left(\mathcal{L}(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}) - \frac{1}{S-1} \sum_{j=1, j \neq i}^{S} \mathcal{L}(\boldsymbol{x}_j, \boldsymbol{b}_j, \boldsymbol{\lambda}) \right) \tag{7}$$

The above is an unbiased estimator and implies no additional cost beyond the S samples.We also propose to use Quasi-Monte Carlo (QMC) sampling [32] for variance reduction. QMC replaces S randomly drawn samples by a pseudo-random sequence of samples of length S with low discrepancy. This sequence covers the underlying design space more evenly than the random samples, thereby reducing the variance of the gradient estimator. Fig. (1) numerically shows the advantage of using variance reduction technique. We observe that the variance of the gradient is decreased using the variance reduction methods, specifically in high dimensions where we observe $\sim 10\times$ benefit.

2.4 Multi-fidelity

The main computational bottleneck of the gradient estimation using Eq. (6) is the multiple evaluation of the objective function. This becomes a significant concern for computationally expensive simulators. We propose to solve this problem using the multi-fidelity (MF) method [33]. Suppose we are given a set of L functions modeling the same quantity and arranged in ascending order of accuracy and computational cost $\{f_1, f_2, \ldots, f_L\}$. We want to optimize the design parameter with respect to the highest-fidelity model (f_L). We can estimate the gradient of the corresponding upper bound using the method suggested in [34] as shown below:

$$\frac{\partial U_L}{\partial \theta} \approx \sum_{\ell=1}^{L} \frac{1}{S_\ell} \sum_{i=1}^{S_\ell} \left(\mathcal{L}_\ell(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}) - \mathcal{L}_{\ell-1}(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}) \right) \frac{\partial}{\partial \boldsymbol{\theta}} \log q\left(\boldsymbol{x}_i \mid \theta\right) \tag{8}$$

where S_ℓ is the number of samples used in the estimator at level ℓ and $\mathcal{L}_0(\cdot) = 0$. We want to use more samples from the low-fidelity model and lesser samples as we increase the fidelity. The method to calculate the number of samples is discussed in [34].

2.5 Implementation details

In the present study, we use `PyTorch` [35] to efficiently compute the gradient. After the gradient estimation, we use the ADAM optimizer [36] as the stochastic gradient descent method. In this work, $q(\boldsymbol{x} \mid \boldsymbol{\theta})$ takes the form of a Gaussian distribution with parameters $\boldsymbol{\theta} = \{\mu, \sigma\}$ representing mean and variance. Our proposed algorithms is summarized in Algorithm 1. The source code will be made available upon publication.

3 Numerical Illustrations

We discuss the numerical results of the proposed (MF)Scout-Nd algorithm on sphere-problem of varying dimensions ($d = \{2, 4, 8, 16, 32\}$). We use the data profiles proposed in [8] to compare (MF)Scout-Nd with cBO [19] and COBYLA [20], which are standard derivative-free optimization methods that can handle constraints. We consider the following optimization problem with *noisy* objective:

$$\min_{\boldsymbol{x}} \mathbb{E}_b \left[\sum_{i=1}^{d} x_i^2 + b \right]; \quad s.t. \quad \mathcal{C}(\boldsymbol{x}) \leq 0 \tag{9}$$

Algorithm 1 Stochastic constrained optimization for non-differentiable objective (Scout-Nd)

```
1:  Input: Objective function(s), constraint(s), distribution $q(\boldsymbol{x} \mid \boldsymbol{\theta})$, stochastic gradient descent
    optimizer $\mathcal{G}$ and its hyper-parameters $\eta$, list of penalty terms $\{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_K\}$, $\boldsymbol{\lambda}_K \to \infty$
2:  $\boldsymbol{\theta}_0^0 \leftarrow$ choose starting point, $k \leftarrow 1$
3:  do
4:      $n \leftarrow 0$
5:      do
6:          $\boldsymbol{x}_i \sim q(\boldsymbol{x} \mid \boldsymbol{\theta}_k^n), \quad \boldsymbol{b}_i \sim p(\boldsymbol{b})$            ▷ Sampling step
7:          Evaluate augmented objectives $\mathcal{L}(\boldsymbol{x}_i, \boldsymbol{b}_i, \boldsymbol{\lambda}_k)$            ▷ Eq. (2)
8:          Monte-Carlo gradient estimate $\nabla_{\boldsymbol{\theta}} U$            ▷ Eq. (7), 8
9:          $\boldsymbol{\theta}_k^{n+1} \leftarrow \mathcal{G}(\boldsymbol{\theta}_k^n, \eta, \nabla_{\boldsymbol{\theta}} U)$ and $n \leftarrow n + 1$            ▷ Stochastic Gradient Descent
10:     while $\|\boldsymbol{\theta}_k^n - \boldsymbol{\theta}_k^{n-1}\| > \varepsilon_\theta$
11:     $\boldsymbol{\theta}_{k+1}^0 \leftarrow \boldsymbol{\theta}_k^n$; $\{\mu, \sigma\} \leftarrow \boldsymbol{\theta}_k^n$ and $k \leftarrow k + 1$
12: while $\|\sigma\| > \varepsilon_\sigma$            ▷ Collapse to Dirac-delta
13: return $\{\mu, \sigma\}$
```

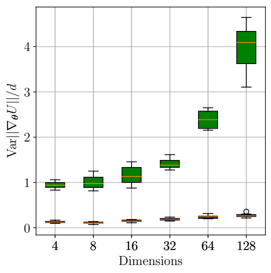


Figure 1: Box plot of the variance of the gradient estimator w.r.t the dimensions with 10 repeated runs for Eq. (9). Gradient estimated with 128 samples at $\boldsymbol{\theta} = \{\boldsymbol{1}_d, e^{\boldsymbol{1}_d}\}$. Green : no variance reduction, red : variance reduction

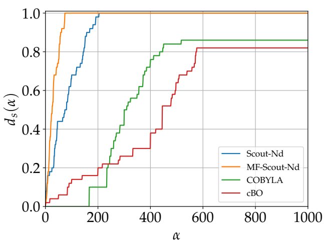


Figure 2: Data profiles plot for the set of optimizers. For the expectation estimation for each step, MF-Scout-Nd uses $S_1 = 10$ HF and $S_2 = 50$ LF evaluations. Every other method uses 50 black-box evaluations. $\epsilon_f = 0.1$

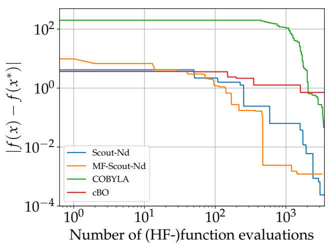


Figure 3: Evolution of the distance between the optimal objective values suggested by the optimizer ($f(x)$) and the theoretical optimum ($f(x^*)$) with respect to the number of (HF-)function evaluations for the first case of sphere problem 9 with $d = 8$.

with $b \sim \mathcal{N}(0, 0.1)$. We consider two different constraint cases. The first case is $\mathcal{C}(\boldsymbol{x}) = 1 - (x_1 + x_2)$ where the optimum lies on the constraint i.e. $\boldsymbol{x}^* = \{0.5, 0.5\} \cup \{0\}_{d-2}$. In the second case, the constraint surface is defined as $\mathcal{C}(\boldsymbol{x}) = \sum_{i=1}^{d} x_i - 1$ leading to the optimum at $\boldsymbol{x}^* = \{0\}_d$. Let $\mathcal{S}$ be the set of optimizers s and $\mathcal{P}$ be a set of benchmark problems p. Then the data profile $d_s(\alpha)$ [8] of a optimizer $s \in \mathcal{S}$ is given by

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} : \frac{t_{p,s}}{d_p + 1} \leq \alpha \right\} \right| \tag{10}$$

where d_p in the number of design variables in p, α is the allowed number of functional evaluations scaled by the number of design variables and $t_{p,s}$ is the minimum number of function calls a solver s requires to reach the optimum of a problem p within accuracy level ϵ_f. We run each benchmark 5 times leading to $|\mathcal{P}| = 5(|d|) \times 5(\text{number of runs}) \times 2(\text{number of cases}) = 50$. For (MF)Scount-Nd, we consider two levels of multi-fidelity. The high-fidelity (HF) is given by the Eq. (9) and the low-fidelity (LF) is defined by down-scaling the x_i in Eq. (9) to have $x_i = x_i/1.05$.

We can observe from Fig. (2) that Scout-Nd performs better than cBO and COBYLA because it solved most of the benchmark problems $p \in \mathcal{P}$ for a given α. Our proposed algorithm outperforms the other two derivative-free methods because we use derivative approximation to move toward the optimum. This not only helped us to converge faster but also tackled high-dimensionality. MF-Scout-Nd performed better as it converged faster towards the optimum than Scout-Nd because it needs fewer costly function evaluations. We can also observe from Fig. (3) that Scout-Nd and MF-Scout-Nd come closer to the actual optimal objective for a given computational budget.

4 Conclusions

We extended the method proposed by [18, 17] to account for constraints. We further employed a multi-fidelity strategy and gradient variance reduction schemes to reduce the number of costly simulator calls. We demonstrated on a classical benchmark problem that our method performs better than the chosen baselines in terms of quality of optimum and number of functional calls. As future work, we will test our method on real-world problems (for example: [15, 14]).

5 Broader Impact Statement

Many real-world systems in engineering and physics are modeled by complex simulators that might be parameterized by a high-dimensional random variable. Some notable examples include particle physics, fluid mechanics, molecular dynamics, protein folding, cosmology, material sciences, etc. Frequently, the simulators are black-box, making the task of optimization/inference challenging, specifically in high dimensions. The task can be further complicated with the inclusion of constraints. In the present work, we introduced an algorithm to approximate the gradients for an optimization/inference task involving these simulators. We demonstrated that the proposed method performs better than the state-of-the-art on a standard benchmark problem.

We do not see any direct ethical concerns associated with this research. The impact on society is primarily through the over-arching context of providing novel methods to approach optimization/inference involving complex simulators.

Acknowledgements

A.A. and P.S.K acknowledge the support of VDI Technologiezentrum Gmbh and the Federal Ministry for Education and Research (BMBF) in the context of the project "Probabilistic machine learning for the calibration and validation of physics-based models of concrete" (FKZ-13XP5125B). K.R and H.J.B acknowledge the support of Helmholtz Association under the research school Munich School for Data Science (MUDS).

References

[1] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. "The frontier of simulation-based inference". In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30055–30062.

[2] German Ros et al. "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3234–3243.

[3] Jonas Degrave, Michiel Hermans, Joni Dambre, et al. "A differentiable physics engine for deep learning in robotics". In: *Frontiers in neurorobotics* (2019), p. 6.

[4] Atul Agrawal and Phaedon-Stelios Koutsourelakis. *A probabilistic, data-driven closure model for RANS simulations with aleatoric, model uncertainty*. 2023. arXiv: `2307.02432 [physics.flu-dyn]`.

[5] Filipe de Avila Belbute-Peres et al. "End-to-end differentiable physics for learning and control". In: *Advances in neural information processing systems* 31 (2018).

[6] Didier Lucor, Atul Agrawal, and Anne Sergent. "Simple computational strategies for more effective physics-informed neural networks modeling of turbulent natural convection". In: *Journal of Computational Physics* 456 (2022), p. 111022.

[7] Rajesh Ranganath, Sean Gerrish, and David Blei. "Black box variational inference". In: *Artificial intelligence and statistics*. PMLR. 2014, pp. 814–822.

[8] Jorge J Moré and Stefan M Wild. "Benchmarking derivative-free optimization algorithms". In: *SIAM Journal on Optimization* 20.1 (2009), pp. 172–191.

[9] Wolfgang Banzhaf et al. *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc., 1998.

[10] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems* 25 (2012).

[11] Friedrich Menhorn et al. "A trust-region method for derivative-free nonlinear constrained stochastic optimization". In: *arXiv preprint arXiv:1703.04156* (2017).

[12] Shakir Mohamed et al. "Monte carlo gradient estimation in machine learning". In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5183–5244.

[13] Georg Ch Pflug. *Optimization of stochastic models: the interface between simulation and optimization*. Vol. 373. Springer Science & Business Media, 2012.

[14] Gilles Louppe, Joeri Hermans, and Kyle Cranmer. "Adversarial Variational Optimization of Non-Differentiable Simulators". en. In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. ISSN: 2640-3498. PMLR, Apr. 2019, pp. 1438–1447. URL: `https://proceedings.mlr.press/v89/louppe19a.html` (visited on 11/22/2022).

[15] Sergey Shirobokov et al. "Black-box optimization with local generative surrogates". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14650–14662.

[16] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. "Learning to simulate". In: *arXiv preprint arXiv:1810.02513* (2018).

[17] Thomas Bird, Julius Kunze, and David Barber. *Stochastic Variational Optimization*. arXiv:1809.04855 [cs, stat]. Sept. 2018. URL: `http://arxiv.org/abs/1809.04855` (visited on 11/08/2022).

[18] Joe Staines and David Barber. *Variational Optimization*. arXiv:1212.4507 [cs, stat]. Dec. 2012. URL: `http://arxiv.org/abs/1212.4507` (visited on 11/08/2022).

[19] Jacob R Gardner et al. "Bayesian optimization with inequality constraints." In: *ICML*. Vol. 2014. 2014, pp. 937–945.

[20] Michael JD Powell. *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.

[21] Aharon Ben-Tal and Arkadi Nemirovski. "Robust solutions of uncertain linear programs". In: *Operations research letters* 25.1 (1999), pp. 1–13.

[22] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. "Theory and applications of robust optimization". In: *SIAM review* 53.3 (2011), pp. 464–501.

[23] I.-J. Wang and J.C. Spall. "Stochastic optimization with inequality constraints using simultaneous perturbations and penalty functions". en. In: *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*. Maui, HI, USA: IEEE, 2003, pp. 3808–3813. ISBN: 978-0-7803-7924-4. DOI: 10.1109/CDC.2003.1271742. URL: http://ieeexplore.ieee.org/document/1271742/ (visited on 10/28/2022).

[24] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[25] Anthony V Fiacco and Garth P McCormick. *Nonlinear programming: sequential unconstrained minimization techniques*. SIAM, 1990.

[26] Mark A Beaumont, Wenyang Zhang, and David J Balding. "Approximate Bayesian computation in population genetics". In: *Genetics* 162.4 (2002), pp. 2025–2035.

[27] Paul Marjoram et al. "Markov chain Monte Carlo without likelihoods". In: *Proceedings of the National Academy of Sciences* 100.26 (2003), pp. 15324–15328.

[28] Joe Staines and David Barber. "Optimization by Variational Bounding." In: *ESANN*. 2013.

[29] Peter W Glynn. "Likelihood ratio gradient estimation for stochastic systems". In: *Communications of the ACM* 33.10 (1990), pp. 75–84.

[30] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3 (1992), pp. 229–256.

[31] Wouter Kool, Herke van Hoof, and Max Welling. "Buy 4 REINFORCE Samples, Get a Baseline for Free!" en. In: (July 2022). URL: https://openreview.net/forum?id=r1lgTGL5DE (visited on 11/11/2022).

[32] Josef Dick, Frances Y Kuo, and Ian H Sloan. "High-dimensional integration: the quasi-Monte Carlo way". In: *Acta Numerica* 22 (2013), pp. 133–288.

[33] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. "Survey of multifidelity methods in uncertainty propagation, inference, and optimization". In: *Siam Review* 60.3 (2018), pp. 550–591.

[34] Michael B Giles. "Multilevel monte carlo methods". In: *Acta numerica* 24 (2015), pp. 259–328.

[35] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[36] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

E
Stochastic Black-Box Simulator Optimization using Multi-Fidelity Score Function Estimator

MACHINE
LEARNING
Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
18 July 2024

REVISED
23 October 2024

ACCEPTED FOR PUBLICATION
1 November 2024

PUBLISHED
31 January 2025

PAPER

Stochastic black-box optimization using multi-fidelity score function estimator

Atul Agrawal[1,3,*], **Kislaya Ravi**[2,3], **Phaedon-Stelios Koutsourelakis**[1] **and Hans-Joachim Bungartz**[2]

[1] Professorship of Data-Driven Materials Modelling,Technical University of Munich, Munich, Germany
[2] School of Computation, Information and Technology, Technical University of Munich, Munich, Germany
[3] Equal contribution.
[*] Author to whom any correspondence should be addressed.

E-mail: atul.agrawal@tum.de, kislaya.ravi@tum.de, p.s.koutsourelakis@tum.de and bungartz@tum.de

Keywords: black-box optimization, multi-fidelity, score function estimator, windfarm layout optimization, optimization under uncertainty

Abstract

Optimizing parameters of physics-based simulators is crucial in the design process of engineering and scientific systems. This becomes particularly challenging when the simulator is stochastic, computationally expensive, black-box and when a high-dimensional vector of parameters needs to be optimized, as e.g. is the case in complex climate models that involve numerous interdependent variables and uncertain parameters. Many traditional optimization methods rely on gradient information, which is frequently unavailable in legacy black-box codes. To address these challenges, we present SCOUT-Nd (Stochastic Constrained Optimization for N dimensions), a gradient-based algorithm that can be used on non-differentiable objectives. It can be combined with natural gradients in order to further enhance convergence properties. and it also incorporates multi-fidelity schemes and an adaptive selection of samples in order to minimize computational effort. We validate our approach using standard, benchmark problems, demonstrating its superior performance in parameter optimization compared to existing methods. Additionally, we showcase the algorithm's efficacy in a complex real-world application, i.e. the optimization of a wind farm layout.

1. Introduction

Several real-world continuous optimization problems are too difficult to solve due to the involvement of complex physics-based simulators in the objective or the constraints. These physics-based simulators are used across fields of engineering and science to drive research [1]. There are many problems that fit into this category, spanning a wide array of fields such as aeronautic design [2, 3], applications in healthcare and science [4, 5], material design [6, 7], optimizing particle-physics instruments [8, 9], optimizing wind-farm layouts [10], accelerator physics [11], reinforcement learning (RL) [12] and generating synthetic training data for machine learning related tasks [13, 14].

We consider a high-dimensional and expensive to a query function $f:\mathbb{R}^d \rightarrow \mathbb{R}$. The high-dimensional parameter space, multiple local optima, the lack of gradients, and stochasticity in the objective function evaluation make the optimization non-trivial. This setting is prevalent in science and engineering domains [15]. Optimization is primarily facilitated through gradient-based or gradient-free ('black-box') methods [16]. Gradient-based methods are effective when derivatives are readily available [5, 17–19]. Over the past decade, the surge in interest in machine learning (ML) has significantly propelled the field of differentiable programming [16]. However, in most real-world optimization scenarios involving physics-based simulators, gradients are not available inherently [20]. As an example, consider the case of the optimization of a wind farm's layout with respect to the total power output. Numerical simulations based on well-established, computational fluid dynamics models that account for turbine interactions and wake effects, can provide accurate predictions but as is the case for most legacy codes, they do not provide gradients. More importantly, they are very computationally demanding and the cost of their repeated solution during any search procedure can be gigantic. The gradients are typically obtained through one of several strategies: (a) employing the adjoint method [5, 21, 22], (b) rewriting the simulator in a differentiable programming language such as JAX, PyTorch, Julia, etc [23, 24], or (c) developing a differentiable surrogate, e.g. based on neural networks, to approximate the objective function (e.g. [9]). The first two options pose significant implementation challenges, often involving extensive modifications to legacy codes, thus impeding applying ML to scientific computing [25]. Furthermore, training a differentiable surrogate can be prohibitively expensive due to the initial cost of data generation. Also, the effectiveness of the optimization is contingent upon the quality of the surrogate model developed. Recently, automatic differentiable (AD) compiler methods [26] have emerged, providing gradients of legacy codes written in C, C++, Fortran, etc. These methods facilitate integration with existing ML infrastructure and support gradient-based optimization. However, these approaches are not directly comparable to the methods proposed in the current study, as compiler-based methods assume legacy codes are compiled on low level virtual machine-based compilers, which may not always be satisfied.

The gradient-free methods [27–29] are used for optimization when only black-box evaluations are possible. It is also commonly called simulation-based inference/optimization [1, 30]. Some widely used methods include genetic algorithms [31], Bayesian Optimization and their extensions [32, 33], and Evolution strategy (ES) methods [34–36], to name a few. The gradient-free methods perform poorly on high-dimensional parametric spaces [29]. To remedy this, stochastic gradient estimators [37] have been recently used to estimate gradients of black-box functions, by employing a differentiable surrogate (e.g. in the form of deep neural network) and, hence, perform gradient-based optimization [8, 13, 38–42]. However, they do not account for the constraints. Also, these strategies are continent of the quality of the surrogate model learned. The quality of the surrogate model is dependent on the architecture of the ML model and availability of the training data (which can be a bottleneck in high-dimensional cases, as data requirements grow linearly with parametric dimensions e.g. [8]). Furthermore, since the methods rely on Monte Carlo techniques, the computational cost can be very high due to many expensive simulator calls, and the gradients can be very noisy.

This work introduces a novel approach SCOUT-Nd (Stochastic Constrained Optimization for *N* Dimensions) and MF-SCOUT-Nd (Multi-Fidelity Stochastic Constrained Optimization for *N* Dimensions) for constrained, stochastic optimization involving black-box simulators with high-dimensional, parametric dependency. As per the notion of oracles [16, 43], the proposed algorithm uses a stochastic, zeroth-order oracle [44]. We draw inspiration from Variational Optimization [45, 46] to estimate the gradients, provide extensions that account for constraints, and employ multi-fidelity (MF) strategies to improve efficiency by incorporating lower-fidelity and less expensive simulator(s). The proposed algorithm consists of the following major elements: (a) A non-intrusive method to estimate gradients of black-box physical simulators (section 2.2), with an ability to account for stochasticity in the objective (section 2) and handle constraints using penalty methods (section 2.1). (b) Strategies to reduce the variance of the gradient estimator (section 2.3). (c) Ability to handle non-convexity (section 2.6). (d) More efficient and robust convergence properties using natural gradients (section 2.4). (e) Multi-fidelity strategies (section 2.7) and adaptive selection of the number of samples for gradient estimation (section 2.6.4 and section 2.7.1) in order to reduce computational cost while retaining predictive accuracy. The structure of the rest of the paper is as follows. Section 2 defines the problem we address in the present work and the proposed algorithm with relevant details and analysis. In section 3, we present the state-of-the-art performance of SCOUT-Nd/MF-SCOUT-Nd on standard benchmark analytical problems and compare the results with popular (constrained) optimization methods like constrained Bayesian optimization (cBO)[47], COBYLA [48], DYCORS [49] and SLSQP [50]. Secondly, we test our algorithms on a real-world, expensive, physics-based simulator. We choose the windfarm layout optimization case [15], which, owing to the absence of derivatives, presence of constraints, multi-model surface, and stochasticity, provides a challenging test case for black-box optimization routines [51]. We compare our results with SLSQP. In section 4, we summarize our findings and discuss limitations and potential enhancements.

2. Methodology

Problem statement. We are concerned with optimizing a scalar-valued function $f(\boldsymbol{x},\boldsymbol{b})$ subject to constraints $\boldsymbol{\mathcal{C}}(\boldsymbol{x}) = \{\mathcal{C}_1(\boldsymbol{x}),\ldots,\mathcal{C}_I(\boldsymbol{x})\}$, where $\boldsymbol{x}\in\mathbb{R}^d$ denote the potentially high-dimensional deterministic parameters and $\boldsymbol{b}\sim p(\boldsymbol{b})$ represents uncertain parameters [38], which act as stochastic inputs to the function f. A lack of knowledge about the parameters or the inherent noise in the system may cause uncertainty. The objective f

or the constraints $\mathcal{C}$ depend implicitly on the output of the black-box simulator, thus we only have access to a zero-order oracle. Since the solution of the optimization problem to obtain the optimal design $\boldsymbol{x}$ involves the random vector $\boldsymbol{b}$, its variability needs to be involved in the optimization process to limit its negative effect on the optimal design. This is classically done using a robustness measure [52, 53] given by $\mathcal{R}$. The general parameter-dependent nonlinear constrained optimization problem can be stated as

$$\min_{\boldsymbol{x}} \mathcal{R}\left[f(\boldsymbol{x},\boldsymbol{b})\right], \quad \text{s.t} \quad \mathcal{C}_i\left(\boldsymbol{x}\right) \leqslant 0, \ \ \forall i \in \{1,\ldots,I\}. \tag{1}$$

In this work, we will only consider the expectation as a robustness measure, in which case, the problem can be stated as follows:

$$\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}}\left[f(\boldsymbol{x},\boldsymbol{b})\right], \quad \text{s.t} \quad \mathcal{C}_i\left(\boldsymbol{x}\right) \leqslant 0, \ \ \forall i \in \{1,\ldots,I\}. \tag{2}$$

In addition to that, the gradient of the objective function and the constraint is unavailable. Hence, one cannot directly apply gradient-based optimization methods.

2.1. Penalizing the constraints

To tackle the constrained optimization problem (equation (2)), we convert it to an unconstrained one using penalty-based methods [54, 55]. We define an augmented objective function $\mathcal{L}$ as follows:

$$\mathcal{L}\left(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}\right) = f(\boldsymbol{x},\boldsymbol{b}) + \sum_{i=1}^{I} \lambda_i \max\left(\mathcal{C}_i\left(\boldsymbol{x}\right),0\right), \tag{3}$$

where $\lambda_i > 0$ is the penalty parameter for the *ith* constraint and the $\max(\cdot,\cdot)$ controls the magnitude of the penalty applied. One can make the enforcement of a particular constraint stricter by increasing the value of the corresponding penalty parameter. Incorporating the augmented objective (equation (3)) in equation (2), one can arrive at the following optimization problem:

$$\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}}\left[\mathcal{L}\left(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}\right)\right]. \tag{4}$$

Optimization. The Monte Carlo method approximates the expectation, which induces noise. We alleviate the dependence on the penalty parameter $\boldsymbol{\lambda}$ by using the sequential unconstrained minimization technique (SUMT) algorithm [56]. The algorithm considers a strictly increasing sequence $\{\boldsymbol{\lambda}_n\}$ with $\boldsymbol{\lambda}_n \to \infty$. [56] show that when $\boldsymbol{\lambda}_n \to \infty$, then the sequence of corresponding minima, say $\{\boldsymbol{x}_n^*\}$, converges to a global minimizer $\boldsymbol{x}^*$ of the original constrained problem. The SUMT technique has also been shown to work with non-linear constraints [57]. This adaptation of the penalty parameters helps to balance the need to satisfy the constraints with the need to make progress towards the optimal solution. Augmented Lagrangian [58] represents an alternative approach that could be considered for application in this context, though it has not been explored in the current study.

2.2. Gradient estimation

Since the design variable $\boldsymbol{x}$ can be high-dimensional, gradient-free methods such as genetic programming, Bayesian Optimization, etc may not be as efficient as gradient-based, and the latter should be used whenever available [28]. Unfortunately, the direct computation of derivatives of $\mathcal{L}$ with respect to the optimization variables $\boldsymbol{x}$ is not feasible because of the unavailability of the gradients of the objective function and the constraints. One notes many active research threads across disciplines are trying to tackle this bottleneck of unavailability of gradients in physical simulators [1, 5, 8, 39, 59, 60]. We draw inspiration from the Variational Optimization [13, 45, 61], which constructs an upper bound of the objective function as shown below:

$$\min_{\boldsymbol{x}} \int \mathcal{L}\left(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}\right) p\left(\boldsymbol{b}\right) \,\mathrm{d}\boldsymbol{b} \leqslant \int \mathcal{L}\left(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}\right) p\left(\boldsymbol{b}\right) q\left(\boldsymbol{x}\mid\boldsymbol{\theta}\right) \,\mathrm{d}\boldsymbol{b}\,\mathrm{d}\boldsymbol{x} = U(\boldsymbol{\theta}), \tag{5}$$

where $q(\boldsymbol{x}\mid\boldsymbol{\theta})$ is a probability density over the design variables $\boldsymbol{x}$ with parameters $\boldsymbol{\theta}$. If $\boldsymbol{x}^*$ yields the minimum of the objective $\mathbb{E}_{\boldsymbol{b}}[\mathcal{L}]$, then this can be achieved with a degenerate q that collapses to a Dirac-delta, i.e. if $q(\boldsymbol{x}\mid\boldsymbol{\theta}) \approx \delta(\boldsymbol{x}-\boldsymbol{x}^*)$. The inequality above would generally be strict for all other densities q or parameters $\boldsymbol{\theta}$. Hence, instead of minimizing $\mathbb{E}_{\boldsymbol{b}}[\mathcal{L}]$ with respect to $\boldsymbol{x}$, we can minimize the upper bound $U(\boldsymbol{\theta})$ with respect to the distribution parameters $\boldsymbol{\theta}$. Under mild restrictions outlined by [46], the bound $U(\boldsymbol{\theta})$ is differentiable w.r.t $\boldsymbol{\theta}$. One can evaluate the gradient of $U(\boldsymbol{\theta})$ using the 'log-derivative trick' [37] as shown below:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x},\boldsymbol{b}} \left[\mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\right] \\
&= \nabla_{\boldsymbol{\theta}} \int q(\boldsymbol{x} \mid \boldsymbol{\theta})\, p(\boldsymbol{b})\, \mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{b} \\
&= \int \nabla_{\boldsymbol{\theta}} q(\boldsymbol{x} \mid \boldsymbol{\theta})\, p(\boldsymbol{b})\, \mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{b} \\
&= \int q(\boldsymbol{x} \mid \boldsymbol{\theta})\, \nabla_{\theta} \log q(\boldsymbol{x} \mid \boldsymbol{\theta})\, p(\boldsymbol{b})\, \mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{b} \\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{b}} \left[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta})\, \mathcal{L}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\right].
\end{aligned} \tag{6}$$

The Monte Carlo estimation of the expectation shown in equation (6) is as follows:

$$\frac{\partial U}{\partial \boldsymbol{\theta}} \approx \frac{1}{S} \sum_{i=1}^{S} \mathcal{L}(\boldsymbol{x}_i,\boldsymbol{b}_i,\boldsymbol{\lambda})\, \frac{\partial}{\partial \boldsymbol{\theta}} \log q(\boldsymbol{x}_i \mid \boldsymbol{\theta}). \tag{7}$$

Equation (6) is known as the score function estimator [62]. In a manner similar to the REINFORCE [12], we take gradient steps on $\boldsymbol{\theta}$. The score function estimator also appears in the context of RL[63]. In the present work, we work with functions with continuous domains, so we use Gaussian for $q(\cdot)$. For the special case where $q(\boldsymbol{x} \mid \boldsymbol{\theta})$ is factored Gaussian, the resulting gradient estimator is also known as *parameter-exploring policy gradients* [64], or *zero-order gradient estimation* [65]. The number of samples S per iteration is usually of the order $\mathcal{O}(d)$ [35, 63] (also demonstrated in section 2.6.4). This can be problematic for high-dimensional problems. Fortunately, the gradient estimation can be embarrassingly parallelized. One needs to synchronize random seeds between machines before optimization, i.e. each machine knows what perturbations the other machine used, so each machine only needs to communicate a single scalar to and from the other machine to agree on a parameter update. This approach also ensures that gradient estimation is non-intrusive, meaning that the physics-based simulator does not need modification to accommodate the optimization routine. Consequently, legacy simulators can be used without any adjustments.

2.3. Variance reduction

Monte Carlo gradient estimation suffers from high variance. Extensive work has been done to address the issue of high variance in the past decades [37]. In the present work, we propose using the baseline method discussed in [66] to reduce the mean square error (MSE) of the estimator in equation (7). This is given by:

$$\frac{\partial U}{\partial \boldsymbol{\theta}} \approx \frac{1}{S} \sum_{i=1}^{S} \frac{\partial}{\partial \boldsymbol{\theta}} \log q(\boldsymbol{x}_i \mid \boldsymbol{\theta}) \left(\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{b}_i,\boldsymbol{\lambda}) - \frac{1}{S-1} \sum_{j=1, j\neq i}^{S} \mathcal{L}\left(\boldsymbol{x}_j,\boldsymbol{b}_j,\boldsymbol{\lambda}\right) \right). \tag{8}$$

The above is an unbiased estimator with no additional cost beyond the S samples.

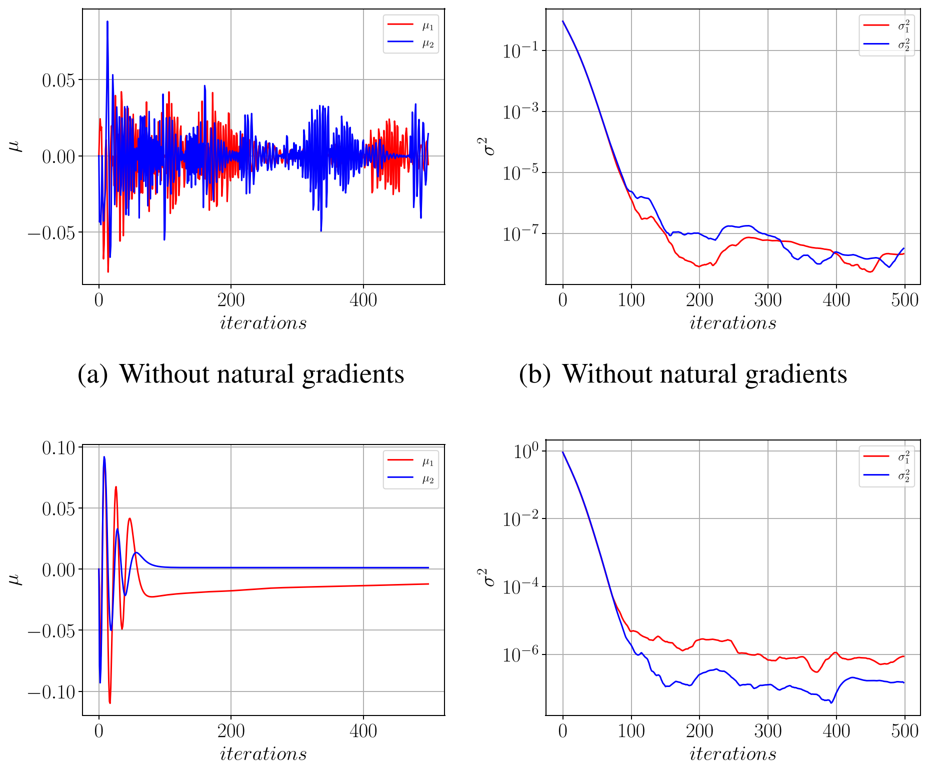
We also propose to use Quasi-Monte Carlo (QMC) sampling [67] for variance reduction, thus promising a more accurate gradient estimate [68]. In our previous work [42], we showed numerically the advantage of using the combination of QMC and the baseline strategy. QMC replaces S randomly drawn samples with a pseudo-random sequence of samples of length S with low discrepancy. This sequence covers the underlying design space more evenly than the random samples, thereby reducing the variance of the gradient estimator. Also, it has been shown that under certain conditions, QMC ($\mathcal{O}(S^{-1})$) reaches a faster rate of convergence as compared to random sampling ($\mathcal{O}(S^{-1/2})$). From a theoretical point of view, the benefit of QMC vanishes in very high dimensions. However, [69] showed that the gains are observed up to dimension 150 in practice. In this work, we present the results using Sobol points [70] and sample points with randomized QMC to ensure unbiaseness.

2.4. Natural gradients

With the estimated gradient $\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$, using any gradient descent scheme, one can update the parameters $\boldsymbol{\theta}$ of the distribution $q(\boldsymbol{x}|\boldsymbol{\theta})$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_{\text{step}} \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}), \tag{9}$$

where η_{step} is a learning rate parameter. However, [34] reported that this update is not *scale-invarient* as the gradient controls both the position and variance of a distribution over the same search space dimension and reported unstable parametric updates. When we numerically investigate this using the ADAM optimizer and



(a) Without natural gradients (b) Without natural gradients

(c) With natural gradients (d) With natural gradients

Figure 1. Illustrations highlighting the effect of natural gradients on the 2D Ackley function optimization near the optima.

equation (8), we also observe parametric oscillations in the optimization of the 2D Ackley function around the optima, as can be seen in figures 1(a) and (b). The Ackley function is widely used to study optimization algorithms due to its complex, multi-modal landscape. These oscillations occur due to moving in the parametric space $\boldsymbol{\theta}$ using the gradient descent with Euclidean distance as the distance measure. To alleviate this instability, [71, 72] reported that step size reduction with increasing iterations is crucial for stability. To make the parametric updates invariant w.r.t. the particular parametrization used and adaptively select step size, [34] suggests employing *natural gradients*. Natural gradients [71] have been shown numerous advantages over *plain gradients*, e.g. natural gradients showcase better convergence behavior in optimization landscapes with ridges and plateaus. So, in this work, we utilize natural gradients, which relies on a more 'natural' measure of distance $D_{\mathrm{KL}}(\boldsymbol{\theta}'\|\boldsymbol{\theta})$ between the distributions $q(\boldsymbol{x}|\boldsymbol{\theta})$ and $q(\boldsymbol{x}|\boldsymbol{\theta}')$, with $D_{\mathrm{KL}}(\boldsymbol{\theta}'\|\boldsymbol{\theta})$ denoting the Kullback-Leibler divergence [73]. The natural gradient is then the solution to a constrained optimization problem which involves (for details, please see [34])

$$\boldsymbol{F} = \int q(\boldsymbol{x} \mid \boldsymbol{\theta})\, \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta})^{\top}\, \mathrm{d}\boldsymbol{x} \tag{10}$$

$$= \mathbb{E}_{\boldsymbol{x}} \left[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta})\, \nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta})^{\top} \right], \tag{11}$$

where $\boldsymbol{F}$ is the *Fisher Information Matrix* (FIM) of the given parametric family of search distributions. As mentioned earlier, in SCOUT-Nd/MF-SCOUT-Nd, the distribution q takes the form of a multivariate normal (MVN) with $\boldsymbol{\theta} := (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma} := \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$. Using the analytical derivative of the MVN, we obtain the following expression for the FIM:

$$\boldsymbol{F}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathrm{diag}\left(\frac{1}{\sigma_1^2}, \cdots, \frac{1}{\sigma_d^2}, \frac{2}{\sigma_1^2}, \cdots, \frac{2}{\sigma_d^2} \right). \tag{12}$$

We use $\sigma_i^2 = e^{2\beta_i}$, reparametrized FIM is given by:

$$\boldsymbol{F}(\boldsymbol{\mu}, \boldsymbol{\beta}) = \mathrm{diag}\left(e^{-2\beta_1}, \cdots, e^{-2\beta_d}, 2\boldsymbol{I}_d \right). \tag{13}$$

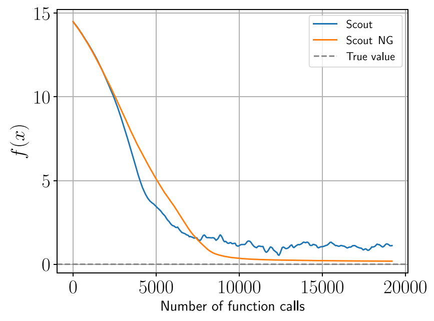


Figure 2. Evolution of Ackley function value with $d = 64$ to highlight the influence of natural gradients. Theoretical $f(\boldsymbol{x}^*) = 0$, learning rate $= 0.1$, number of samples $= 64$.

The analytical derivation of the above expression is provided in appendix A. If the $\boldsymbol{F}$ matrix is invertible, the Monte Carlo gradient estimate is updated to

$$\tilde{\nabla}_{\boldsymbol{\theta}} U = \boldsymbol{F}^{-1} \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) \,. \tag{14}$$

The gradient in the equation above is called the *natural gradient* [71, 74]. As can be seen in the equation above, the $\boldsymbol{F}$ matrix scales the gradients. Therefore, it can also be seen as a step-size adaptation.

The tendency of the natural gradient is to reduce the value of the gradient if the variance is smaller than one. This slows down the convergence speed in the beginning, even after dampening. One needs natural gradients only near the optimum. So, we apply the natural gradients only when the optimizer is near the optimum. We can identify if the optimizer is in the proximity of the optimum when the value of the variance has decreased to a small value (i.e. $\|\boldsymbol{\Sigma}\| \leqslant \sigma_{\text{nat-grad}}$). Another way suggested by [72] is to apply a dampening constant to the Fisher information matrix, given by

$$\tilde{\boldsymbol{F}} = \boldsymbol{F} + \eta \boldsymbol{I}. \tag{15}$$

The isotropic damping η can be set constant or using an exponential decay approach by

$$\eta = \begin{cases} \tilde{\eta} & \text{if iteration } k < N_{\text{cut-off}}, \\ \max\left(\tilde{\eta} \cdot \exp\left(-\frac{k - N_{\text{cut-off}}}{N_{\text{cut-off}}}\right), \eta_{\text{lower bound}}\right) & \text{else}. \end{cases} \tag{16}$$

In subsequent numerical experiments, the parametric values choose are $\tilde{\eta} = 10^{-1}$, $\eta_{\text{lower bound}} = 10^{-6}$ and $N_{\text{cut-off}} = 100$.

Using the Ackley function, we study the benefits of adding natural gradients to the proposed algorithm. As can be seen in figures 1(c) and (d), the evolution of the parameters $\boldsymbol{\theta}$ around the optimum are without oscillations, as opposed to the case when natural gradients are not used (figures 1(a) and (b)). This well-behaved parametric evolution also translates to a better optimum, even in high dimensional cases, as seen in figure 2.

2.5. Termination criterion

Our proposed algorithms use gradient descent methods to optimize the objective. Gradient-based optimization methods converge to a local optimum. Subsequent optimization steps yield only marginal improvements. In such scenarios, continuing the optimization process may be inefficient. It can be beneficial to terminate the algorithm to conserve computational resources. This section explores several criteria for terminating the algorithm, addressing both constrained and unconstrained cases separately.

In unconstrained optimization, no penalty term is involved, simplifying the optimization algorithm to a single loop that repeatedly estimates gradients and takes gradient descent steps. We terminate when the computational budget exceeds a predefined limit or the distribution q collapses to a Dirac-delta. For a Gaussian distribution, this occurs when the norm of the covariance falls below a specified cut-off value $\sigma_{\text{cut-off}}$. The method proposed for unconstrained optimization problems is summarized in the Algorithm 1.

There are two loops in the algorithm for constrained optimization. The outer loop increases the value of the penalty constant ($\boldsymbol{\lambda}$). The inner loop performs the gradient descent step for a fixed penalty constant. We terminate the outer loop under the same conditions as the unconstrained optimization i.e. when the computational budget runs out or the distribution q collapses to a Dirac-delta. For the inner loop, termination occurs when the computational budget exceeds the predefined limit for a penalty constant or the

Algorithm 1. SCOUT-Nd algorithm for unconstrained optimization.

```
input: Objective function $f(\boldsymbol{x},\boldsymbol{b})$ , distribution $q(\boldsymbol{x}\mid\boldsymbol{\theta})$, gradient descent optimizer $\mathcal{G}$, gradient descent step size $\eta_{\text{step}}$,
       Natural gradient starting variance $\sigma_{\text{nat-grad}}$, Maximum budget $N_{\max}$, variance cut-off value $\sigma_{\text{cut-off}}$
1 Set initial point $\boldsymbol{\theta}_0 := \{\boldsymbol{\mu}_0,\boldsymbol{\Sigma}_0\}$
2 Initialize $k \leftarrow 0$
3 do
4     $\boldsymbol{x}_i \sim q(\boldsymbol{x}\mid\boldsymbol{\theta}_k), \quad \boldsymbol{b}_i \sim p(\boldsymbol{b})$          // Sampling step
5     Evaluate objective function $f(\boldsymbol{x}_i,\boldsymbol{b}_i)$
6     Monte Carlo gradient estimate $\nabla_{\boldsymbol{\theta}} U$          // equations (8) and (7)
7     if $\|\boldsymbol{\Sigma}_k\| < \sigma_{\text{nat-grad}}$ then
8         Compute the natural gradients $\tilde{\nabla}_{\boldsymbol{\theta}} U$          // equation (14)
9         $\nabla_{\boldsymbol{\theta}} U \leftarrow \tilde{\nabla}_{\boldsymbol{\theta}} U$
10    end
11    $\boldsymbol{\theta}_{k+1} \leftarrow \mathcal{G}(\boldsymbol{\theta}_k,\nabla_{\boldsymbol{\theta}} U,\eta_{\text{step}})$          // Gradient Descent
12    $k \leftarrow k+1$
13 while $k < N_{\max}$ and $\|\boldsymbol{\Sigma}_k\| > \sigma_{\text{cut-off}}$
output $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$
```

Algorithm 2. SCOUT-Nd algorithm for constrained optimization.

```
input: Objective function $f(\boldsymbol{x},\boldsymbol{b})$ , constraints $\boldsymbol{\mathcal{C}}(\boldsymbol{x})$, distribution $q(\boldsymbol{x}\mid\boldsymbol{\theta})$, gradient descent optimizer $\mathcal{G}$, gradient
       descent step size $\eta_{\text{step}}$, list of penalty terms $\{\boldsymbol{\lambda}_m\}_{m=1}^M$, $\boldsymbol{\lambda}_M \to \infty$, Natural gradient starting variance $\sigma_{\text{nat-grad}}$,
       Maximum budget for inner and outer loop $N_{\max}^{\text{inner}}$, $N_{\max}^{\text{outer}}$, variance cut-off value $\sigma_{\text{cut-off}}$
1 Set initial point $\boldsymbol{\theta}_0 := \{\boldsymbol{\mu}_0,\boldsymbol{\Sigma}_0\}$
2 Initialize $k \leftarrow 0$, $m \leftarrow 1$
3 do
4     $n \leftarrow 0$
5     do
6         $\boldsymbol{x}_i \sim q(\boldsymbol{x}\mid\boldsymbol{\theta}_k), \quad \boldsymbol{b}_i \sim p(\boldsymbol{b})$          // Sampling step
7         Evaluate augmented objectives $\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{b}_i,\boldsymbol{\lambda}_m)$          // equation (3)
8         Monte Carlo gradient estimate $\nabla_{\boldsymbol{\theta}} U$          // equations (8) and (7)
9         if $\|\boldsymbol{\Sigma}_k\| < \sigma_{\text{nat-grad}}$ then
10            Compute the natural gradients $\tilde{\nabla}_{\boldsymbol{\theta}} U$          // equation (14)
11            $\nabla_{\boldsymbol{\theta}} U \leftarrow \tilde{\nabla}_{\boldsymbol{\theta}} U$
12        end
13        $\boldsymbol{\theta}_{k+1} \leftarrow \mathcal{G}(\boldsymbol{\theta}_k,\nabla_{\boldsymbol{\theta}} U,\eta_{\text{step}})$          // Gradient Descent
14        $n \leftarrow n+1, k \leftarrow k+1$
15    while $k < N_{\max}^{\text{outer}}$ and $\|\boldsymbol{\Sigma}_k\| > \sigma_{\text{cut-off}}$
16    if $\exists i : \mathcal{C}_i(\boldsymbol{\mu}_k) > 0$ and $\|\boldsymbol{\Sigma}_k\| > \sigma_{\text{cut-off}}$ then
17        $\boldsymbol{\Sigma}_k \leftarrow \boldsymbol{\Sigma}_0$
18    end
19    $m \leftarrow m+1$
20 while $n < N_{\max}^{\text{inner}}$ and $\|\boldsymbol{\Sigma}_k\| > \sigma_{\text{cut-off}}$
output $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$
```

distribution q collapses to a Dirac-delta. Additionally, we might encounter cases where, for a fixed value of $\boldsymbol{\lambda}$, the optimizer gets stuck at a local optimum in a region where the constraint is not satisfied because the penalty constant is insufficient. This situation can be detected if the norm of the variance falls below a specified cut-off value ($\sigma_{\text{cut-off}}$) and the constraint is not satisfied. In such cases, we reset the value of the covariance term ($\boldsymbol{\Sigma}_k$) to its initial value. The method proposed for constrained optimization problems is summarized in the algorithm 2.

2.6. Method analysis

2.6.1. Covergence studies

Our proposed algorithm optimizes $U(\boldsymbol{\theta})$ instead of $f(\boldsymbol{x})$. Let us convert the distribution to standard normal distribution as follows:

$$U(\boldsymbol{\theta}) = \mathbb{E}\left[f(\boldsymbol{\mu}+\boldsymbol{\sigma}\cdot\boldsymbol{z})\right]_{\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}, \tag{17}$$

where $\boldsymbol{\sigma}$ is the vector containing the diagonal entries of $\boldsymbol{\Sigma}$. We are ignoring the stochastic term $\boldsymbol{b}$ from the analysis because we assume that the noise term gets marginalized in equation (17). Using the Taylor expansion and the simplifications detailed in the appendix C, we get

$$U(\boldsymbol{\theta}) = f(\boldsymbol{\mu}) + \frac{1}{2}\sum_{i=1}^{d} {\sigma_i}^2 \frac{\partial^2 f}{\partial {x_i}^2}(\boldsymbol{\mu}) + O\left(\|\boldsymbol{\sigma}\|^4\right). \tag{18}$$

Taking the derivative of equation (18) with respect to $\boldsymbol{\mu}$:

$$\frac{\partial U}{\partial \boldsymbol{\mu}} = \frac{\partial f}{\partial \boldsymbol{\mu}}(\boldsymbol{\mu}) + \frac{1}{2}\sum_{i=1}^{d} {\sigma_i}^2 \frac{\partial^3 f}{\partial {x_i}^2 \partial \boldsymbol{\mu}}(\boldsymbol{\mu}) + O\left(\|\boldsymbol{\sigma}\|^4\right). \tag{19}$$

The mean $\boldsymbol{\mu}$ corresponds to the location $\boldsymbol{x}$ in the parameter space. We can ignore the higher order variance term in equation (19) for $\|\boldsymbol{\sigma}\| < 1$. So, the difference between $\partial U/\partial \boldsymbol{\mu}$ and $\partial f/\partial \boldsymbol{x}$ has a term involving the variance and the third derivative of the function. Let us call that term as *drift term*. In an ideal case scenario, the derivative of the objective function with respect to the parameter $\partial f/\partial \boldsymbol{x}$ should be equal to the derivative of the convoluted objective function with respect to the mean of the distribution $\partial U/\partial \boldsymbol{\mu}$. Let the optimum parameter obtained using the gradient descent algorithm performed on U be $\boldsymbol{\theta}^* = \{\boldsymbol{\mu}^*, \boldsymbol{\sigma}^*\}$ and on f be $\boldsymbol{x}^*$. If we ensure that $\partial f/\partial \boldsymbol{x} = \partial U/\partial \boldsymbol{\mu}$, the $\boldsymbol{\mu}^*$ and $\boldsymbol{x}^*$ should converge to the same value. This is achievable when either $\|\boldsymbol{\sigma}\| \to 0$ or the third derivative term tends to zero.

Now, taking the derivative of equation (18) with respect to $\boldsymbol{\sigma}$ and ignoring the higher order terms of $\boldsymbol{\sigma}$, we get

$$\frac{\partial U}{\partial \sigma_i} = \sigma_i \frac{\partial^2 f}{\partial x_i^2}. \tag{20}$$

The second derivative of a function is positive at a local minimum. If the function f does not have a high rate of change of curvature, then the derivative of U with respect to σ_i is greater than zero $(\partial U/\partial \sigma_i > 0)$ in the proximity of the local minimum, Therefore, the gradient descent method reduces the value of $\|\boldsymbol{\sigma}\|$ as it approaches the local minimum. From equation (18), we can observe that the value of the drift term becomes smaller as $\|\boldsymbol{\sigma}\| \to 0$, leading to $U(\boldsymbol{\mu}, \boldsymbol{\sigma}) \to f(\boldsymbol{x})$. Under this condition, $\boldsymbol{\mu}^* \to \boldsymbol{x}^*$ and $\boldsymbol{\sigma}^* \to 0$, implying that the optimum of the convoluted function U approaches the local optimum of the original objective function f. Additionally, we can utilize $\|\boldsymbol{\sigma}\| \to 0$ as a convergence criterion for the method.

2.6.2. Gradient correction for constraints

Let us consider an optimization problem:

$$\min_{x} x^2, \quad \text{s.t} \quad 1 - x \leqslant 0. \tag{21}$$

The optimum for the optimization problem in equation (21) lies at 1.0, which coincides with the boundary of the feasible domain. The figure 3 shows the plot corresponding to the constraint element of the augmented function $\mathcal{L}(x)$, as well as the constraint segment within the upper bound $U(\mu, \sigma)$ over a range of variance values for the distribution. Ideally, the constraint term should not exert any influence within the feasible region. The figure 3 reveals that the constraint term's influence on $U(\mu, \sigma)$ remains nonzero within the feasible zone. This influence progressively diminishes with the reduction in the σ value. Notably, this influence tends to deter the optimizer from maintaining proximity to the constraint boundary, even when the optimum resides along this boundary. When we run our suggested algorithm to solve the problem in equation (21), the algorithm will reach the area inside the feasible domain and get pushed away from the boundary because of the constraints. Eventually, the value of σ decreases, and the effect of the constraint around the boundary diminishes, moving towards convergence at the boundary. This approach is suboptimal, mainly when dealing with computationally demanding objective functions, as it decelerates the convergence rate. We propose a strategy aimed at mitigating this effect by enforcing the value of the gradient of the constraint's contribution to the upper bound (U) with respect to the position parameter $(\boldsymbol{\mu})$ to assume a zero value if that constraint is satisfied $\left(\text{i.e., } \frac{\partial U_{\mathcal{C}_j}}{\partial \boldsymbol{\mu}} = 0, \quad \text{if} \quad \mathcal{C}_j(\boldsymbol{\mu}) \leqslant 0, \text{where } U_{\mathcal{C}_j} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{b}}[\mathcal{C}_j(\boldsymbol{x}, \boldsymbol{b})]\right)$.

The derivative of the augmented function is the sum of the derivatives of the objective function and the penalty term. The penalty term has zero gradient in the feasible region. However, the score function estimator can estimate a non-zero gradient in the vicinity of the feasible region. By using this heuristic, we assist our optimizer by providing known information. It helps the optimizer reach the optimum more efficiently. This heuristic is applied only when the optimizer is in the vicinity of the boundary of the feasible region.

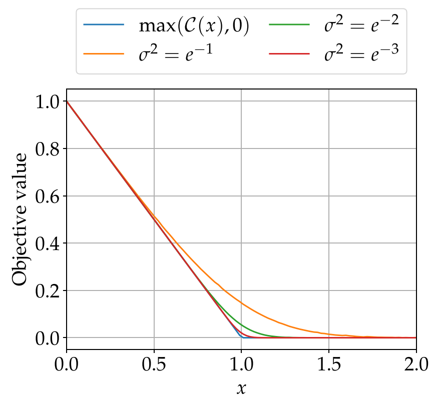


Figure 3. Contribution of the constraint term for the problem given in equation (21). The blue curve shows the constraint term in the augmented function $\mathcal{L}(x)$. The orange, green, and red curves show the constraint term in the upper bound $U(\mu, \sigma)$ for $\sigma^2 = e^{-1}$, $\sigma^2 = e^{-2}$ and $\sigma^2 = e^{-3}$ respectively. We observe that the tendency of the constraint to move the optimum away from the boundary is higher for a bigger value of σ. This misguides the optimizer, especially when the optimum lies on the boundary of the feasible region.

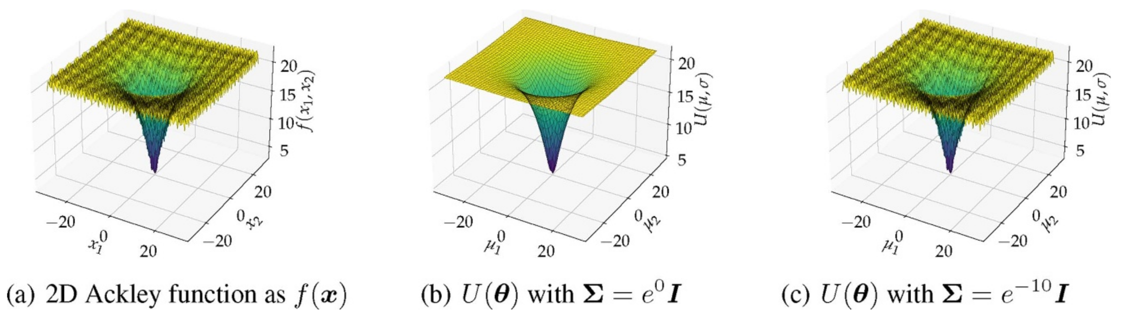


Figure 4. Comparing the effect of Gaussian convolution on the 2D Ackley function.

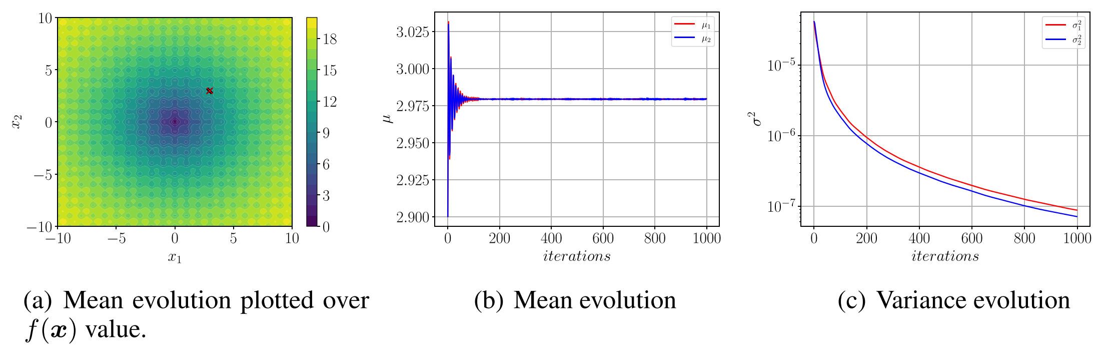


Figure 5. Effect of starting the optimization with a very small variance. The starting value of variance is given as $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1^2, \sigma_2^2)$ with $\sigma_1^1 = \sigma_2^2 = e^{-10}$.

2.6.3. Overcoming multiple local optima

Casting the objective $f(\boldsymbol{x})$ as $U(\boldsymbol{\theta})$ has the additional advantage of escaping local optima by smoothing the objective function. It can be viewed as a Gaussian-blurred version of the original objective $f(\cdot)$, free of non-smoothness. The degree of this smoothing is contingent upon the choice of $\boldsymbol{\Sigma}$. Examplalarily, we demonstrate this smoothening effect using the Ackley function, which has a highly non-convex surface, as illustrated in figure 4. The figure hints towards the significant effect the initial value of the design variable variance can have on the quality of the optimum. We optimized the 2D Ackley function using our proposed algorithm under two distinct initial variance scenarios to study the effect of initial design variable variance. This is illustrated in figures 5 and 6 for small and high variance, respectively. The optimizer is trapped in a local minimum when the initial variance is small (reference figure 5) and dodges several local optima to converge at the global optima (reference figure 6) when the initial variance is high. This property is also

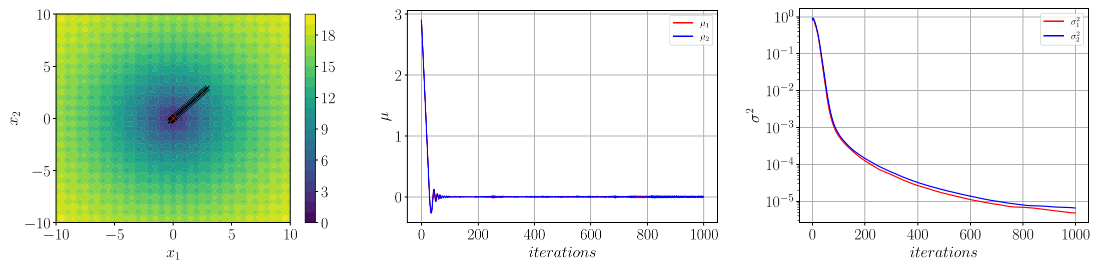


Figure 6. Effect of starting the optimization with a high variance. The starting value of variance is given as $\boldsymbol{\Sigma} = \text{diag}(\sigma_1^2, \sigma_2^2)$ with $\sigma_1^1 = \sigma_2^2 = e^0$.

confirmed upon testing using the Rastringen function (reference figure 23). The variance parameter controls the degree of smoothness of the convoluted loss manifold. Without knowing the level of non-convexity apriori in the loss landscape, it is difficult to comment on the initial value of variance required. A robust choice would be starting with a relatively high value of the variance, though this might lead to higher total computational cost.

2.6.4. Sample size

Let us assume that after kth step of optimization, we are in a state $\boldsymbol{\theta}_k := (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, where $\boldsymbol{\mu}_k := \{\mu_{1,k}, \ldots, \mu_{d,k}\}$ and $\boldsymbol{\Sigma}_k := \text{diag}(\sigma_{1,k}^2, \ldots, \sigma_{d,k}^2)$. Let us represent the exact and the Monte Carlo approximation of the gradient in equation (7) by $\nabla_{\boldsymbol{\theta}_k} U$ and $\widehat{\nabla_{\boldsymbol{\theta}_k} U}$ respectively. Let us represent the variance of the samples used to approximate the gradient in equation (7) at the kth optimization step by $\boldsymbol{V}_k := \{V_{1,k}, \ldots, V_{d,k}, \ldots, V_{2d,k}\}$ with

$$V_{i,k} = \mathbb{V}\left[\mathcal{L}(\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{\lambda}) \frac{\partial}{\partial \theta_i} \log q(\boldsymbol{x} \mid \boldsymbol{\theta}_k)\right]. \tag{22}$$

One can write the MSE of the gradient approximation as

$$\mathbb{E}\left[\left(\widehat{\nabla_{\boldsymbol{\theta}_k} U} - \nabla_{\boldsymbol{\theta}_k} U\right)^2\right] = \frac{\boldsymbol{V}_k}{S}. \tag{23}$$

Upon using the vanilla stochastic gradient descent method with learning rate α, the next exact state ($\boldsymbol{\theta}_{k+1}$), and the approximate state ($\widehat{\boldsymbol{\theta}}_{k+1}$) can be obtained as

$$\begin{aligned} \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k + \alpha \nabla_{\boldsymbol{\theta}_k} U, \\ \widehat{\boldsymbol{\theta}}_{k+1} &= \boldsymbol{\theta}_k + \alpha \widehat{\nabla_{\boldsymbol{\theta}_k} U}. \end{aligned} \tag{24}$$

Note that this analysis can be extended to other variations of the gradient descent methods by changing the formula in equation (24). We can write the MSE of the state as

$$\mathbb{E}\left[\left(\widehat{\boldsymbol{\theta}}_{k+1} - \boldsymbol{\theta}_{k+1}\right)^2\right] = \alpha^2 \frac{\boldsymbol{V}_k}{S}. \tag{25}$$

The Monte Carlo estimator in equation (7) is unbiased i.e. $\mathbb{E}[\widehat{\nabla_{\boldsymbol{\theta}_k} U}] = \nabla_{\boldsymbol{\theta}_k} U$. This transfers the unbiasedness to the states, which leads to

$$\mathbb{E}\left[\widehat{\boldsymbol{\theta}}_{k+1}\right] = \boldsymbol{\theta}_{k+1}. \tag{26}$$

Now, we want to choose the number of samples such that distance between the Gaussian distribution obtained by exact gradient value ($q(\boldsymbol{\theta}_{k+1})$) and the Gaussian distribution obtained by the approximate gradient ($q(\widehat{\boldsymbol{\theta}}_{k+1})$) in probability space is less than a desired value given by ε_{KL}. This distance measure is given by the Kullback–Leibler divergence [75], written as

$$D_{\mathrm{KL}}\left(q\left(\widehat{\boldsymbol{\theta}}_{k+1}\right)||\,q\left(\boldsymbol{\theta}_{k+1}\right)\right) = \frac{1}{2}\left(\log\frac{|\boldsymbol{\Sigma}|}{|\widehat{\boldsymbol{\Sigma}}|} + tr\left(\boldsymbol{\Sigma}^{-1}\widehat{\boldsymbol{\Sigma}}\right) + \left(\widehat{\boldsymbol{\mu}}-\boldsymbol{\mu}\right)^T\boldsymbol{\Sigma}^{-1}\left(\widehat{\boldsymbol{\mu}}-\boldsymbol{\mu}\right) - d\right),$$
$$= \frac{1}{2}\sum_{i=1}^{d}\left(\log\sigma^2_{i,k+1} - \log\widehat{\sigma}^2_{i,k+1} + \frac{\widehat{\sigma}^2_{i,k+1}}{\sigma^2_{i,k+1}} + \frac{\left(\widehat{\mu}_{i,k+1}-\mu_{i,k+1}\right)^2}{\sigma^2_{i,k+1}}\right) - \frac{d}{2}, \tag{27}$$

where $\widehat{(\cdot)}$ represent the states obtained using approximate gradients. Taking expectation on both sides of equation (27), then simplifying the expression using equations (25), and (26), we obtain

$$\mathbb{E}\left[D_{\mathrm{KL}}\left(q\left(\widehat{\boldsymbol{\theta}}_{k+1}\right)||\,q\left(\boldsymbol{\theta}_{k+1}\right)\right)\right]$$
$$= \frac{1}{2}\sum_{i=1}^{d}\left(\log\sigma^2_{i,k+1} - \mathbb{E}\left[\log\widehat{\sigma}^2_{i,k+1}\right] + \frac{\mathbb{E}\left[\widehat{\sigma}^2_{i,k+1}\right]}{\sigma^2_{i,k+1}} + \frac{\mathbb{E}\left[\left(\widehat{\mu}_{i,k+1}-\mu_{i,k+1}\right)^2\right]}{\sigma^2_{i,k+1}}\right) - \frac{d}{2},$$
$$= \frac{1}{2}\sum_{i=1}^{d}\left(\log\sigma^2_{i,k+1} - \mathbb{E}\left[\log\widehat{\sigma}^2_{i,k+1}\right] + \frac{\alpha^2 V_{i,k}}{S\sigma^2_{i,k+1}}\right), \tag{28}$$
$$< \frac{1}{2}\sum_{i=1}^{d}\left(\log\sigma^2_{i,k+1} - \underbrace{\log\mathbb{E}\left[\widehat{\sigma}^2_{i,k+1}\right]}_{\text{Jensen's ineqality}} + \frac{\alpha^2 V_{i,k}}{S\sigma^2_{i,k+1}}\right) = \sum_{i=1}^{d}\frac{\alpha^2 V_{i,k}}{2S\sigma^2_{i,k+1}} \quad \text{(using equation (26))},$$
$$\Longrightarrow \mathbb{E}\left[D_{\mathrm{KL}}\left(q\left(\widehat{\boldsymbol{\theta}}_{k+1}\right)||\,q\left(\boldsymbol{\theta}_{k+1}\right)\right)\right] < \sum_{i=1}^{d}\frac{\alpha^2 V_{i,k}}{2S\sigma^2_{i,k+1}}$$

We obtained the upper bound of the KL divergence between the approximate and exact distribution. We enforce a bound on the KL divergence by bounding the upper bound by a parameter $\varepsilon_{\mathrm{KL}}$, providing an expression for the sample size

$$\mathbb{E}\left[D_{KL}\left(q\left(\widehat{\boldsymbol{\theta}}_{k+1}\right)||\,q\left(\boldsymbol{\theta}_{k+1}\right)\right)\right] < \varepsilon_{\mathrm{KL}},$$
$$\Longrightarrow \sum_{i=1}^{d}\frac{\alpha^2 V_{i,k}}{2S\sigma^2_{i,k+1}} < \varepsilon_{\mathrm{KL}}, \tag{29}$$
$$\Longrightarrow S > \sum_{i=1}^{d}\frac{\alpha^2 V_{i,k}}{2\varepsilon_{\mathrm{KL}}\sigma^2_{i,k+1}}.$$

When one uses natural gradients then the expression becomes

$$S > \sum_{i=1}^{d}\frac{\alpha^2 V_{i,k}\sigma^4_{i,k}}{2\varepsilon_{\mathrm{KL}}\sigma^2_{i,k+1}}. \tag{30}$$

From the above, we observe:

- For higher dimensional problems, more samples are required.
- Variance reduction techniques help in decreasing sample size requirement by reducing the variance of the gradient $\boldsymbol{V}_k$.
- The number of samples is proportional to the step size. Smaller step sizes require fewer samples because of small changes in the parameter values in the gradient descent step.
- When employing natural gradients, the number of samples required is relatively smaller when closer to the optimum (i.e. $\sigma^4_{i,k}\to 0$).

By using equation (29) or equation (30), we can dynamically determine the number of samples required at each optimization step. We initially sample a predefined number of points and then verify if the conditions are met. If they are unsatisfied, we sample additional points to fulfill the requirements. We iteratively add the newly sampled points, continuing this process until the conditions in equations (29) and (30) are satisfied. To conserve computational resources, we utilize both new and existing points instead of resampling all points. However, this approach precludes using QMC due to the non-nested behavior of Sobol points.

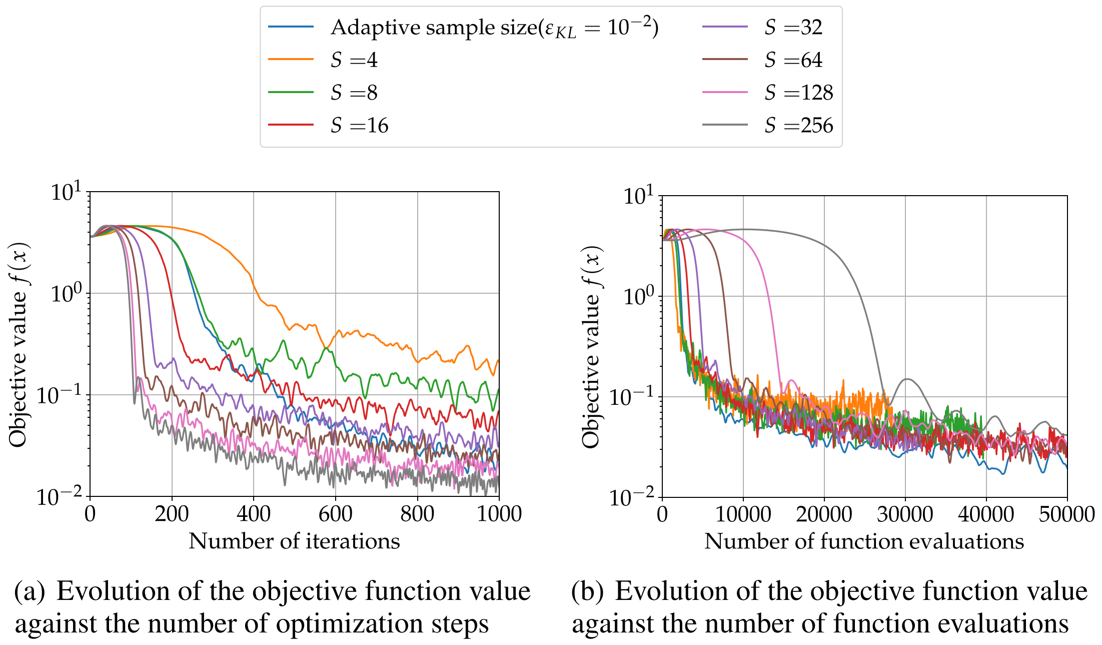


Figure 7. Numerical study to compare the performance SCOUT-Nd with adaptive sample size and fixed sample sizes on 32-dimensional Ackley function. We switch off the natural gradients and QMC for all the cases to ensure a similar condition for all the test cases.

At the start of the kth optimization phase, the exact value of $\boldsymbol{\Sigma}_{k+1}$ is unknown. We tackle this issue using an iterative process. We start the estimation of the number of samples using $\sigma^2_{i,k}$ instead of $\sigma^2_{i,k+1}$ in equation (29). Subsequently, we determine $\boldsymbol{\Sigma}_{k+1}$ via a gradient descent technique and check if the criteria in equation (29) is satisfied. If not, we modify the sample size and recalibrate $\boldsymbol{\Sigma}_{k+1}$ by re-executing the gradient descent step. We repeat this process until the condition in equation (29) is satisfied.

We numerically evaluate the performance of adaptive sample size against various fixed sample size cases in optimizing the 32-dimensional Ackley function. Figure 7 illustrates the evolution of the objective function against the number of optimization steps and function evaluations. As discussed in algorithms 2 and 1, natural gradients are employed when the variance falls below a specified threshold ($\sigma_{\text{nat-grad}}$). Since natural gradients start at different steps for different test cases, they are excluded from this numerical test to ensure fair comparison. Additionally, we do not use QMC methods for fixed sample size cases, as QMC is incompatible with the adaptive sample size method.

From figure 7(a), we observe that the convergence rate improves as the number of samples increases. Additionally, the optimum quality is better with a larger sample size. This is attributed to the smaller MSE in gradient estimation associated with larger sample sizes, leading to faster convergence and improved optimum results. Notably, optimization with adaptive sample sizes falls in between, achieving an optimum quality comparable to that of optimization with 128 samples.

We observe from figure 7(b) that as the number of samples increases, the convergence slows down. While maintaining a low mean squared error (MSE) in each iteration brings advantages, it also leads to more function evaluations. Optimization using a small sample size converges faster in terms of function evaluations. However, the optimum quality is compromised due to the high noise in gradient estimation, resulting in oscillations around the optimum. In contrast, optimization with adaptive sample size selects an appropriate number of samples based on the estimator's variance.

Based on these observations, we can conclude that a higher sample size is suitable when function evaluation is inexpensive or we have sufficient computational resources to parallelize the gradient estimation step. However, in cases where function evaluation is expensive, using an adaptive sample size is more practical.

2.7. Multi-fidelity

The main computational bottleneck of the gradient estimation using equation (7) is the multiple evaluation of the objective function. This becomes a significant concern for computationally expensive simulators. We propose to solve this problem using the MF method [76] in the Scout algorithm, termed as MF-SCOUT-Nd. Suppose we are given a set of L functions modeling the same quantity and arranged in ascending order of

Table 1. Hyperparameters involved in SCOUT-Nd and MF-SCOUT-Nd.

Hyperparameter	Description
$\sigma_{\text{nat-grad}}$	$\|\|\boldsymbol{\Sigma}\|\|$ cut-off value beyond which natural gradients are applied
S	Number of samples for gradient estimation
η_{step}	Step size for the gradient descent optimizer
$N_{\max}$	Maximum number of steps in unconstrained optimization
$\sigma_{\text{cut-off}}$	$\|\|\boldsymbol{\Sigma}\|\|$ cut-off value for optimization loop termination
$N_{\max}^{\text{inner}}$	Maximum number of steps for inner loop (constrained optimization)
$N_{\max}^{\text{outer}}$	Maximum number of steps in total (constrained optimization)
$\{\boldsymbol{\lambda}_m\}_{m=1}^M$	List of penalty terms (constrained optimization)
$\{S_\ell\}_{\ell=1}^L$	List of number of samples for gradient estimation (MF-SCOUT-Nd)

accuracy and computational cost $\{f^{(1)}, f^{(2)}, \ldots, f^{(L)}\}$ and the corresponding augmented functions as $\{\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \ldots, \mathcal{L}^{(L)}\}$. To evaluate the gradients, we are calculating the expectation. Following the methods mentioned in [77], we can write the highest fidelity augmented function as a telescopic sum of the other fidelities:

$$\begin{aligned}\mathcal{L}^{(L)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) &= \mathcal{L}^{(1)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) + \left(\mathcal{L}^{(2)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) - \mathcal{L}^{(1)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\right) + \\ &\quad \ldots + \left(\mathcal{L}^{(L)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) - \mathcal{L}^{(L-1)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\right) \\ &= \mathcal{L}^{(1)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) + \sum_{\ell=2}^{L}\left(\mathcal{L}^{(\ell)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) - \mathcal{L}^{(\ell-1)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\right). \end{aligned} \tag{31}$$

Multiplying with $\nabla_{\boldsymbol{\theta}} \log q(x|\boldsymbol{\theta})$ and taking expection on both the sides:

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{b}}\left[\mathcal{L}^{(L)}\nabla_{\boldsymbol{\theta}}\log q(x|\boldsymbol{\theta})\right] = \mathbb{E}_{\boldsymbol{x},\boldsymbol{b}}\left[\mathcal{L}^{(1)}\nabla_{\boldsymbol{\theta}}\log q(x|\boldsymbol{\theta})\right] + \sum_{\ell=2}^{L}\mathbb{E}_{\boldsymbol{x},\boldsymbol{b}}\left[\left(\mathcal{L}^{(\ell)} - \mathcal{L}^{(\ell-1)}\right)\nabla_{\boldsymbol{\theta}}\log q(x|\boldsymbol{\theta})\right].$$

As per the equation (6), the left-hand side of the equation is the gradient of the highest fidelity function with respect to the distribution parameter $\left(\nabla_{\boldsymbol{\theta}} U^{(L)}(\boldsymbol{\theta})\right)$. All the expectations on the right-hand side of the equation can be estimated using an unbiased Monte Carlo estimator. Now, the equation above is simplified as

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} U^{(L)}(\boldsymbol{\theta}) &\approx \frac{1}{S_1}\sum_{i=1}^{S_1}\mathcal{L}^{(1)}(\boldsymbol{x}_i,\boldsymbol{b}_i,\boldsymbol{\lambda})\frac{\partial}{\partial\boldsymbol{\theta}}\log q(\boldsymbol{x}_i \mid \theta) \\ &\quad + \sum_{\ell=2}^{L}\frac{1}{S_\ell}\sum_{i=1}^{S_\ell}\left(\mathcal{L}^{(\ell)}(\boldsymbol{x}_i,\boldsymbol{b}_i,\boldsymbol{\lambda}) - \mathcal{L}^{(\ell-1)}(\boldsymbol{x}_i,\boldsymbol{b}_i,\boldsymbol{\lambda})\right)\frac{\partial}{\partial\boldsymbol{\theta}}\log q(\boldsymbol{x}_i \mid \theta),\end{aligned} \tag{33}$$

where S_ℓ is the number of samples used in the estimator at the fidelity level ℓ. We assume that the approximation quality between each fidelity improves as we increase the fidelity. So, the number of samples required to approximate the expectation decreases as the fidelity increases (i.e. $S_1 > S_2 > \ldots > S_L$). We replace equation (3) and equation (7) with equations (31) and (33) respectively in the Algorithm 1 and 2, with the rest of the steps remaining the same to handle MF. A list of hyperparameters involved in SCOUT-Nd and MF-SCOUT-Nd is provided in table 1.

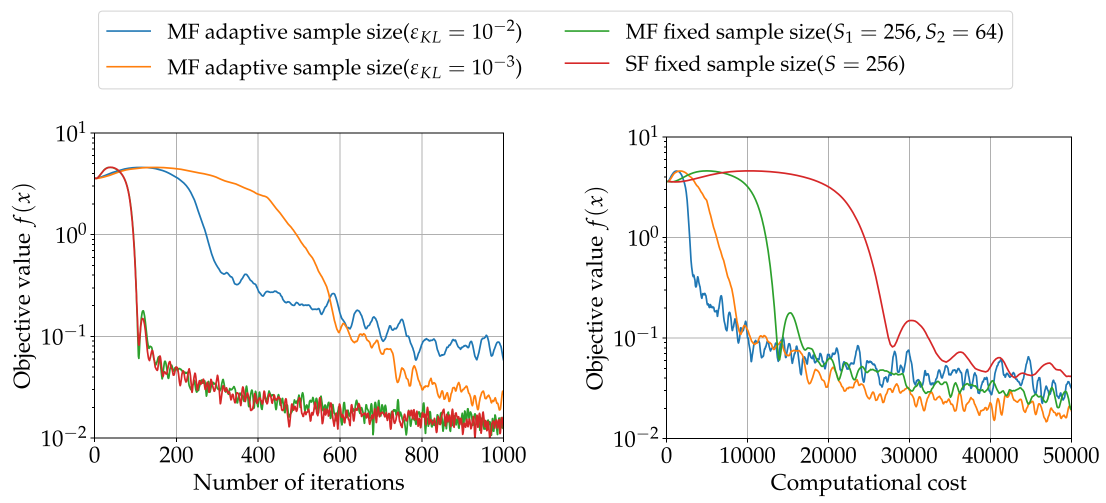
2.7.1. Sample size for MF derivative estimator

In this section, we will discuss the number of samples allocated to each fidelity level for the estimation of the derivative. We follow the approach and notations used in section 2.6.4. Let us define the variable $V_k^{(\ell)}$ as

$$V_k^{(\ell)} = \begin{cases} \sum_{i=1}^{d}\frac{\mathbb{V}\left[\mathcal{L}^{(1)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\nabla_{\theta_i}\log q(\boldsymbol{x}|\boldsymbol{\theta}_k)\right]}{2\sigma_{i,k+1}^2}, & \text{if } \ell = 1, \\ \sum_{i=1}^{d}\frac{\mathbb{V}\left[\left(\mathcal{L}^{(\ell)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda}) - \mathcal{L}^{(\ell-1)}(\boldsymbol{x},\boldsymbol{b},\boldsymbol{\lambda})\right)\nabla_{\theta_i}\log q(\boldsymbol{x}|\boldsymbol{\theta}_k)\right]}{2\sigma_{i,k+1}^2}, & \text{otherwise.}\end{cases} \tag{34}$$

Following the steps in section 2.6.4, we can derive the following for the multi-fidelity case

$$\mathbb{E}\left[D_{\mathrm{KL}}\left(q\left(\widehat{\boldsymbol{\theta}}_{k+1}\right) || \, q(\boldsymbol{\theta}_{k+1})\right)\right] < \sum_{\ell=1}^{L}\frac{\alpha^2 V_k^{(\ell)}}{S_\ell}. \tag{35}$$



(a) Evolution of the objective function value against the number of optimization steps.

(b) Evolution of the objective function value against the computational cost.

Figure 8. Numerical study to compare the performance MF-SCOUT-Nd with adaptive sample size and fixed sample sizes on 32-dimensional Ackley function (the function defined in appendix D). We switch off the natural gradients for all the cases to ensure a similar condition for all the test cases. We assume the high-fidelity model is four times more expensive than the low-fidelity model. We use the high-fidelity function for the single fidelity optimization..

Let C_ℓ be the cost of evaluating $f^{(\ell)}$. We obtain the number of samples for each level by minimizing the total cost such that the upper bound of the KL divergence in equation (35) is equal to $\varepsilon_{\mathrm{KL}}$. This approach is extensively used in multilevel Monte Carlo literature [77–79]. The optimization problem is stated as

$$\begin{aligned} S^*_{\ell,k} =& argmin_{S_\ell} \sum_{\ell=1}^{L} C_\ell S_\ell, \\ \text{s.t.} \quad & \sum_{\ell=1}^{L} \frac{\alpha^2 V_k^{(\ell)}}{S_\ell} = \varepsilon_{\mathrm{KL}}, \end{aligned} \tag{36}$$

where $S^*_{\ell,k}$ represents the optimum number of samples for the ℓth fidelity and kth step. One can analytically solve this problem using a Lagrange multiplier. The optimum number of samples is

$$S^*_{\ell,k} = \left(\frac{\sum_{\ell=1}^{L} \sqrt{V_k^{(\ell)} C_\ell}}{\varepsilon_{\mathrm{KL}}} \right) \alpha^2 \sqrt{\frac{V_k^{(\ell)}}{C_\ell}}. \tag{37}$$

The value of $\boldsymbol{\Sigma}_{k+1}$ is not known when we want to calculate the number of samples. To overcome this problem, we use the same procedure as described in section 2.6.4.

We conducted a numerical evaluation to compare the performance of multi-fidelity gradient estimation using fixed and adaptive sample sizes against single-fidelity fixed sample size cases with 256 high-fidelity (HF) samples to optimize the 32-dimensional Ackley function. Figure 8 illustrates the evolution of the objective function against the number of optimization steps and function evaluations. We did not employ natural gradient and QMC methods for the same reasons discussed in section 2.6.4. We assume the HF model is four times more expensive than the low-fidelity (LF) model.

From figure 8(a), we observe that the convergence rate with respect to the optimization step for the multi-fidelity method with fixed sample size is almost the same as the single-fidelity methods with 256 samples even when multi-fidelity method uses less computational resources demonstrating the advantage of using the multi-fidelity method. Additionally, the multi-fidelity adaptive sample size approach with a smaller upper bound ($\varepsilon_{\mathrm{KL}} = 10^{-3}$) reaches a better the optimum as compared to the bigger upper bound ($\varepsilon_{\mathrm{KL}} = 10^{-2}$) due to a more accurate approximation of the gradients.

We observe from figure 8(b) that multi-fidelity methods converge faster with respect to the computational cost than the single-fidelity method. The multi-fidelity adaptive sample method also outperforms the multi-fidelity fixed sample size method by selecting the optimal number of points based on the estimator's variance while minimizing the computational cost. The case with a larger $\varepsilon_{\mathrm{KL}}$ exhibits a faster initial convergence rate but results in a lower quality optimum. Therefore, it is advisable to begin with a larger value of $\varepsilon_{\mathrm{KL}}$ for faster initial convergence and decrease it as you approach the optimum.

Based on these observations, we conclude that the multi-fidelity methods help reduce computational resources compared to the single-fidelity method, and the adaptive formulation further decreases the computational load. Additionally, one can enhance the optimum quality by adjusting the value of $\varepsilon_{\mathrm{KL}}$.

3. Numerical illustrations

This section compares the proposed algorithm with the state-of-the-art algorithms using standard benchmark analytical problems. Additionally, we highlight the algorithm's efficacy in a complex real-world application: optimizing wind farm layout. In the following, we use `PyTorch` [80] to efficiently compute the gradient of the densities. After the gradient estimation, we use the ADAM optimizer [81] as the stochastic gradient descent method. In this work, $q(\boldsymbol{x} \mid \boldsymbol{\theta})$ takes the form of a Gaussian distribution unless otherwise stated with parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ representing mean and diagonal covariance[4], respectively. The code for the algorithm and the numerical experiments can be found in https://github.com/KislayaRavi/scout-Nd.

3.1. Benchmark Studies

Before applying our algorithm to a real-world problem, we test it on standard optimization benchmarks. We select thirty benchmarks to test the algorithm's ability to handle different types of optimization challenges like multi-modality, constraints, behavior in valleys, dimension scalability, etc. The list of benchmarks is given in the appendix D. We slightly modify the function to make the corresponding LF function for MF-SCOUT-Nd. For each problem, Gaussian noise with a standard deviation of 0.0001 was added to turn the problem stochastic.

We compare our proposed algorithm with state-of-the-art (constrained)optimization algorithms like constrained optimization by linear approximation (COBYLA) [48], sequential least squares programming (SLSQP) [50], and cBO [47]. We use the implementation of COBYLA and SLSQP from the Scipy library [82] and the implementation of cBO from the BayesianOptimization library [83]. In the subsequent experiments, the SLSQP algorithm employed finite difference-based gradient estimation, specifically relying on forward differences to approximate the gradients. Additionally, we also compare our method with DYnamic COordinate search using Response Surface models (DYCORS) [49]. Since DYCORS only supports unconstrained optimization, we compare it by selecting only the unconstrained problem. We use the pySOT [84] implementation of DYCORS. We provide the detailed hyperparameter list of all the optimization methods in appendix E.

We use the data profile curve [29] to compare the overall performance of the algorithms for the whole set of benchmarks. Let $\mathcal{S}$ represent the set of optimizers and $\mathcal{P}$ be a set of benchmark problems. The data profile $d_s(\alpha)$ [29] of an optimizer $s \in \mathcal{S}$ is given by

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} : \frac{t_{p,s}}{d_p + 1} \leqslant \alpha \right\} \right|, \tag{38}$$

where $p \in \mathcal{P}$ represents a benchmark problem, d_p is the dimension of benchmark and $t_{p,s}$ is the minimum number of optimization steps a solver s requires to reach the optimum of a problem p within accuracy level ϵ_f. We ran each benchmark five times, each with different seeds. So, the total number of benchmarks is $|\mathcal{P}| = 30(\text{number of unique benchmarks}) \times 5(\text{number of random seeds}) = 150$. In the benchmark studies, we use a fixed sample size for expectation evaluation for all the optimization methods to ensure an unbiased comparison. The number of LF samples (S_{LF}) and HF samples (S_{HF}) to calculate the expectation at each step of optimization depends upon the dimension of the benchmark problem as shown in table 2.

We can observe from figure 9 that SCOUT-Nd and MF-SCOUT-Nd solved more benchmark problems with a given level of accuracy(ϵ_f). Our proposed algorithms outperform other optimization methods in comparison because we use efficient gradient approximation to move toward the optimum. This not only helped us to converge faster but also tackled high-dimensionality. MF-SCOUT-Nd had a similar performance level as SCOUT-Nd, but it uses less HF function evaluations to reach the same level of accuracy, thereby saving computational resources.

We can observe from figure 10 that all optimizers could reach optima with accuracy level $\epsilon_f = 0.1$. Our constrained optimization problem is convex. So, none of the methods get stuck to the local optima. However, cBO struggles when the accuracy level is $\epsilon_f = 0.01$ because of the high-dimensional constrained problems. Similar to the figure 9, we observe from figure 11 that SCOUT-Nd and MF-SCOUT-Nd outperformed other method by solving most of the unconstrained optimization problems.

[4] In the subsequent investigation $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)$ with $\sigma_i^2 = e^{2\beta_i}$ unless otherwise stated.

Table 2. The table showing the number of low and high-fidelity function evaluations (S_{LF} and S_{HF}) for different benchmark problem (p) as a function of the dimension (d_p).

	$d_p \leqslant 2$		$2 < d_p \leqslant 4$		$4 < d_p \leqslant 8$		$d_p > 8$	
	S_{LF}	S_{HF}	S_{LF}	S_{HF}	S_{LF}	S_{HF}	S_{LF}	S_{HF}
Single-fidelity case	—	32	—	64	—	128	—	256
Multi-fidelity case	32	8	64	16	128	32	256	64

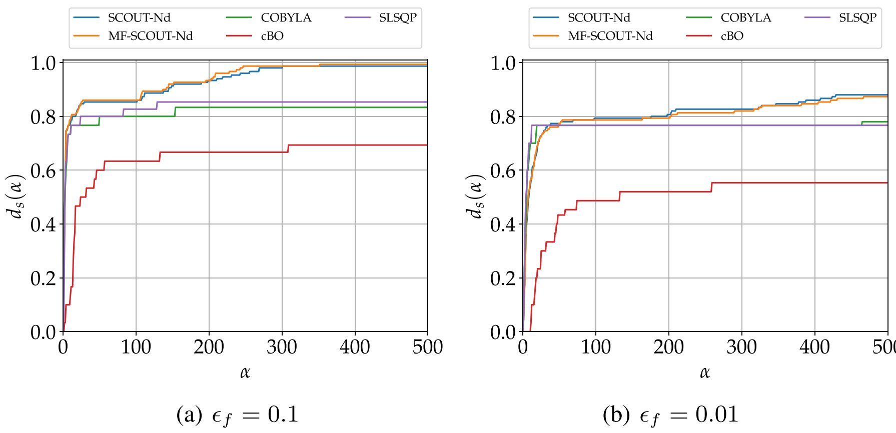


Figure 9. Data Profile plots to show the aggregate performance of the different optimization problems on the given set of benchmark problems. We observe that SCOUT-Nd and MF-SCOUT-Nd were able to solve most of the benchmark problems with both accuracy levels $\epsilon_f = 0.1$, and $\epsilon_f = 0.01$.

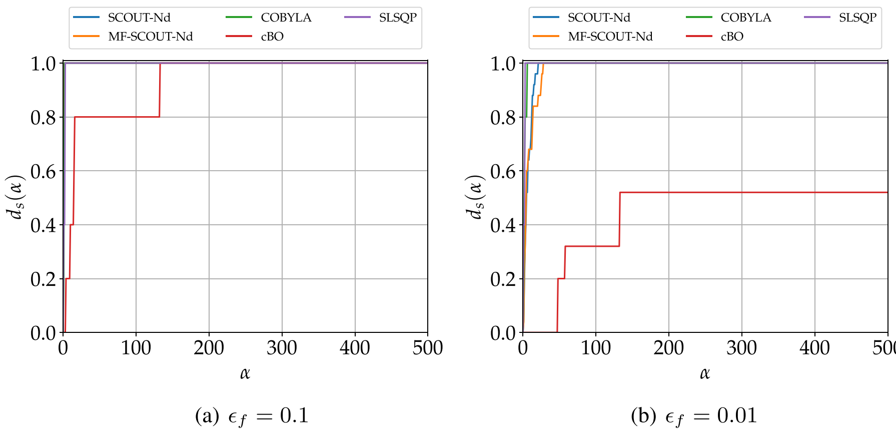
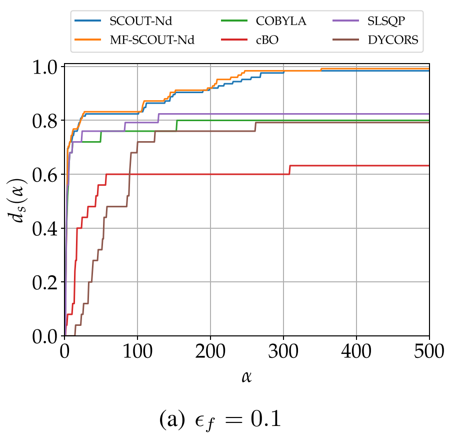


Figure 10. Data Profile plots to show the aggregate performance of the constrained optimization problems. We observe all the methods where able to solve all the problems in accuracy level $\epsilon_f = 0.1$. But, CBO struggles to solve the problems in accuracy level $\epsilon_f = 0.01$.

Our experiments show that COBYLA and SLSQP struggle to handle multimodal models. In some benchmark cases, these two methods converge early to local optima. For instance, we observe this behavior for the 3-dimensional Hartmann function, which has 4 local minima. COBYLA and SLSQP get stuck in one of the local minima as observed in figure 12(a).

Bayesian optimization methods and DYCORS are known to struggle in high-dimensional space [85]. Moreover, they also struggle in the Rosenbrock function, where it finds the valley but takes a long time to converge to the minimum [49, 85]. We plot the evolution of the optimum for the 16-dimensional Rosenbrock function in figure 12(b) and observe that cBO struggles to handle high-dimensionality and valleys.

This section assumes the presence of additive noise as a precondition for ensuring the unbiased estimation of gradients using equation (7). Nonetheless, it is pertinent to acknowledge that applications in real-world scenarios may encounter a variety of noise sources, such as multiplicative noise and parametric noise, among others. We leave the study of different kinds of noise, and its effect on future works.



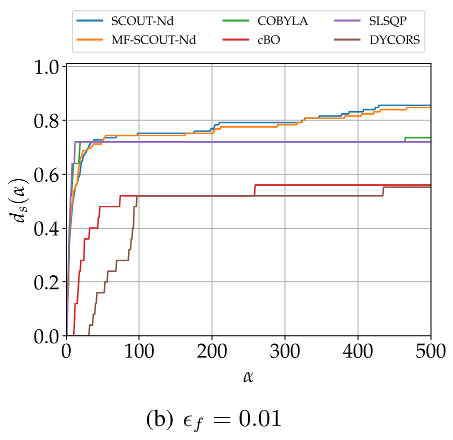
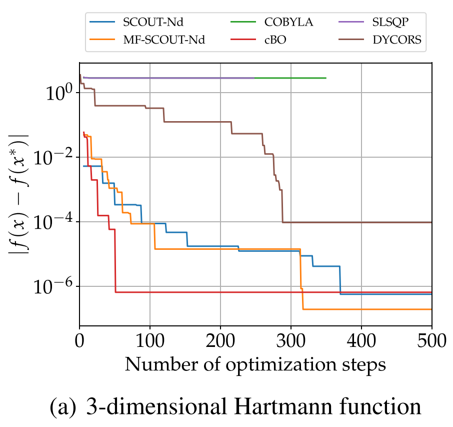


Figure 11. Data Profile plots to show the aggregate performance of the unconstrained optimization problems. We observe that SCOUT-Nd and MF-SCOUT-Nd were able to solve most of the benchmark problems with both accuracy levels $\epsilon_f = 0.1$, and $\epsilon_f = 0.01$.



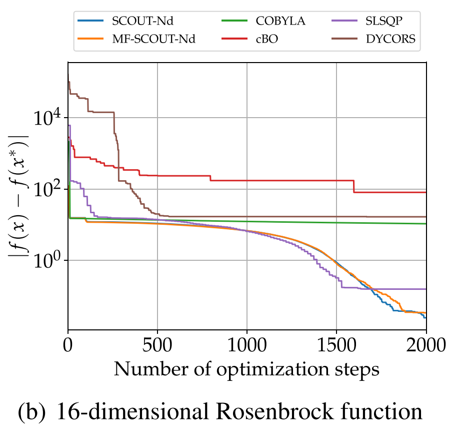


Figure 12. Evolution of optimum for 3-dimensional Hartmann function and 16-dimensional Rosenbrock function. The 3-dimensional Hartmann function has 4 local minima. We observe from figure 12(a) that COBYLA and SLSQP got stuck in a local minimum, whereas other methods that can handle multi-modal problems reached a global minimum. cBO struggles in high-dimensional problems such as 16-dimensional Rosenbrock functions as depicted in figure 12(b).

3.2. Windfarm layout optimization

We apply SCOUT-Nd and MF-SCOUT-Nd to windfarm layout optimization problem [15]. The primary goal of wind farm layout optimization is to position the wind turbines to reduce interference and thus maximize power production. This is a challenging and complex problem due to the variable nature of the wind and the complex interactions between turbines through their wakes. The challenges are listed as follows:

- Uncertainty due to the wind. Generally, the wind speed and wind direction are treated as random variables.
- Availability of only black box evaluations of the numerical simulators [10, 51]. This would make gradient-based methods rely on finite differences, which would significantly increase derivative computational cost and derivatives may also have significant errors [86]. If gradient-free methods are employed, they may struggle when the problem is high dimensional as they do not usually scale well to problems with more than 30 design variables [87].
- A highly multimodal design space, making the problem highly non-convex [88]. This multimodality makes it difficult for classical gradient-based methods, as they require the objective function to be smooth enough [15], or else they might get stuck in local optimum. Gradient-free methods can handle multimodality better, however, they suffer from the curse of dimensionality as discussed above. Thus, there exists no best algorithm, the choice of which is situation and problem-dependent [10, 51].

Owing to the challenges discussed, and their significant overlap with the strengths of the proposed algorithm, the problem serves as an ideal testing ground for SCOUT-Nd/MF-SCOUT-Nd.

Problem definition: The objective of the wind farm layout optimization is to maximize the AEP [5] by changing the position of the wind turbines. The turbines are constrained to stay within a given area and with a minimum separation between them. This objective and the constraints result in a problem of nonlinear optimization under uncertainty with deterministic constraints:

$$\arg\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}}\left[-AEP\left(\boldsymbol{x},\boldsymbol{b}\right)\right] \quad \text{s.t} \quad \boldsymbol{\mathcal{C}}\left(\boldsymbol{x}\right) \leqslant 0, \tag{39}$$

where $\boldsymbol{b} = \{b_1, b_2, \dots, b_n\}$ represents a vector of the random variable for the wind data with $p(\boldsymbol{b})$ being the joint probability density function of the uncertain variables. Common uncertain variables are the wind direction and the freestream wind speed [15]. $\boldsymbol{x}$ denotes a sequence of pairs of coordinates for each wind turbine, which can take on continuous values in a bounded domain. The expected AEP is computed by marginalizing the uncertain parameters. To marginalize, some commonly used methods are Monte Carlo, Polynomial Chaos, rectangular quadrature, etc to name a few [89]. In this work, we used a weighted average, which amounts to the rectangular integration rule. $\boldsymbol{\mathcal{C}}(\boldsymbol{x}) = \{\mathcal{C}_1(\boldsymbol{x}), \mathcal{C}_2(\boldsymbol{x})\}$ denotes the constraints this problem has with $\mathcal{C}_1(\boldsymbol{x})$ denoting the separation between the two turbines constraint and $\mathcal{C}_2(\boldsymbol{x})$ denoting the area constraint. In particular, they are defined as

$$\mathcal{C}_1\left(\boldsymbol{x}\right) = \mathcal{K}\left(2D - Q_{i,j}\right) \quad i,j = 1\dots n_{turbines};\ \ i \neq j \tag{40}$$

$$\mathcal{C}_2\left(\boldsymbol{x}\right) = \mathcal{M}\left(N_{i,m}\right) \quad i = 1\dots n_{turbines};\ \ m = 1\dots n_{boundaries}, \tag{41}$$

where $Q_{i,j}$ is the distance between each pair of turbines i and j , and D is the turbine diameter. The normal distance, $N_{i,m}$, from each turbine i to each boundary m is defined as negative when a turbine is inside the boundary and positive when it is outside the boundary. We aggregate the distance between two turbine constraints using the Kreisselmeier–Steinhauser functional $\mathcal{K}(\cdot)$ [90], which reduces the number of constraints to 1. Also, we define a function $\mathcal{M}(\cdot)$, which aggregates the normal distance constraints by taking the mean of the positive values of $N_{i,m}$. $\mathcal{M}(\cdot)$ is calculated as

$$\mathcal{M}\left(\cdot\right) = \frac{1}{|N^+|}\sum_{\eta_i\in N^+}\eta_i; \qquad \text{with,}\ \ N^+ = \{\eta_i \in N | \eta_i > 0\} \tag{42}$$

where $|N^+|$ is the cardinality of the set N^+. These aggregations produce a less complex optimization problem.

Implementation details/test cases. In the experiments performed, we used the NREL 5MW reference turbine [91]. For wake models, we have used their implementation in FLOw Redirection and Induction in Steady State (FLORIS) [92]. We use Jensen [93][6] as the LF wake model and Gauss–Curl Hybrid (GCH) [94][7] as the HF model as suggested in [95]. The low- and HF models for this problem use different *wind roses*[8] bin resolutions, leading to accuracy and computational differences caused by both fidelity and resolution. Each bin adds an identical set of function calls and operations to the summing process, so the cost scales linearly. In the subsequent experiments, the wind speed is constant at 8 m s^{-1} for simplicity.

We investigate two cases with different numbers of wind turbines, as shown in table 3. The particular choice is made to study the influence of dimensions of the design variable. We use six wind direction bins for the LF and 18 wind direction bins for the HF (reference figure 13). In terms of computational costs, a single LF model evaluation, denoted as c_{LF}, incurs approximately 33% (or one-third) of the cost associated with a single evaluation of the HF model, represented as c_{HF}. For example, in the scenario involving 24 turbines, the Jensen model (the LF model) requires approximately 0.015 s per execution. In contrast, the GCH model(HF model) demands about 0.044 s per execution when run on a single processor core. Consequently, the overall computational expense of determining the annual energy production (AEP) using the HF model is about nine times higher than that required for the LF model.

For both cases, we perform a comparative study between SCOUT-Nd with the HF model, MF-SCOUT-Nd employing both the LF and the HF model, and the SLSQP [50] with the HF model. Within the domain of wind farm layout optimization, variants of sequential quadratic programming (SQP)-based algorithms are predominantly employed [10, 89, 95] due to their gradient-based nature (gradients computed

[5] The Annual energy production (AEP) is given by expected power multiplied by the number of hours in a year.

[6] The Jensen wake model uses a simplistic velocity deficit to represent the wake, and this deficit is summed when wakes interact using the sum-of-squares method.

[7] The solver simplifies the Reynolds-averaged Navier–Stokes equations to obtain a parabolic equation for the wake deficit. The equation is solved in a three-dimensional domain to obtain the wake velocity in a wind plant.

[8] Wind rose is a graphical tool used in the context of wind farms to represent the distribution of wind speed and direction at a particular location.

Table 3. Optimization problem cases: design variables, objective, and constraints.

Cases	objective	design variables	dimension	Constraints
8 turbine farm	AEP	x, y locations	16	Turbine spacing $[252\text{ m}, \infty)$, bound constraint $(x \in [0, 2666\text{ m}], y \in [0, 2666\text{ m}])$
24 turbine farm	AEP	x, y locations	48	Turbine spacing $[252\text{ m}, \infty)$, bound constraint $(x \in [0, 8000\text{ m}], y \in [0, 8000\text{ m}])$

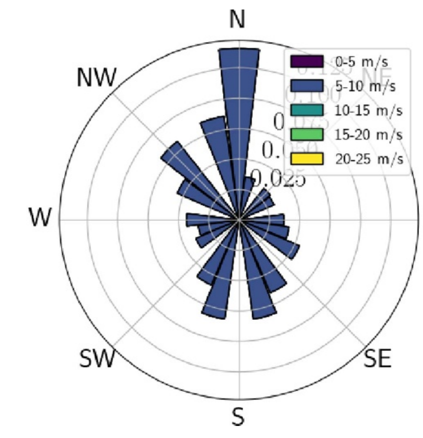


Figure 13. Wind Rose for 18 wind direction bins and a constant wind speed of 8 m s^{-1}.

via finite differences) and their capability to manage constraints effectively. An extensive comparison with different optimization methods is beyond the scope of the present investigation. Interested readers can find details in [10].

Results. This section presents the results obtained upon running the optimization methods for the two cases presented in table 3. The hyper-parameter setting for the optimization methods is given in appendix F, particularly in tables 6 and 7 for the 8 turbines and 24 turbine cases, respectively. Here, we use a fixed sample size for expectation evaluation for all the optimization methods to ensure a fair comparison. For both cases, a grid layout is chosen as the initial layout for all three optimization methods. Naturally, the optimizer is expected to push the turbines towards the boundaries to minimize wake interactions, leading to increased AEP. The results for the two cases with the optimization methods are quantitatively compared in table 4 and table 5, and qualitatively compared in figures 14, 18, 15 and 20. The evolution of the augmented objective, objective and constraints for the optimization with SCOUT-Nd and MF-SCOUT-Nd for 8 turbine case are illustrated in figures 16 and 17 respectively. Similarly, figures 21 and 22 illustrate the evolutions for the 24 turbine case.
We draw the following conclusions from the tables and figures:

- For the 8 turbine case, from figure 14 we observe that all three optimization methods push the turbine towards the boundaries while maintaining the distance between them as per the constraints. This significantly reduces the wake interactions as observed in figure 15. This translates to an increase in AEP as reported in table 4. In this case, SLSQP reports the highest AEP percentage gain (8.73%) from the initial grid layout, followed by SCOUT-Nd (8.26%) and MF-SCOUT-Nd (8.02%). However, the MF-SCOUT-Nd is the cheapest with around 67% computational cost of the SLSQP. This was expected as the SLSQP is a gradient-based method, and dim $= 16$ poses a fairly smooth objective surface, outperforming method like ours. With an optimized value of hyperparameters, SCOUT-Nd/MF-SCOUT-Nd might come closer (or even cross) to the SLSQP in terms of AEP improvement, but this is not the goal of the present investigation.
- For the 24 turbine case, from figure 18 we observe that both SCOUT-Nd and MF-SCOUT-Nd push the turbine towards the boundaries from the initial grid layout, which is expected from global optima. This can also be understood from the power vs. wind direction plot in figure 19. Additionally, figure 20 illustrates the wake interactions along the most prominent wind direction, i.e. from the north side. However, as seen in the figure 18, the SLSQP failed to push the turbine towards the boundaries as it got trapped in local optima. We also confirm this behavior by running the SLSQP optimizer using different initial layouts drawn from

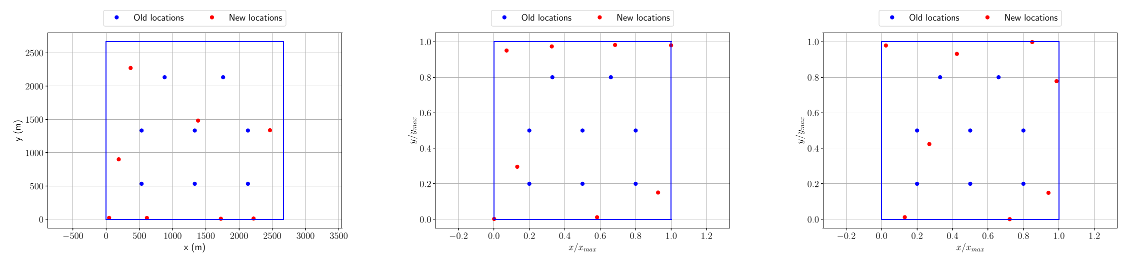
Table 4. Optimization results comparison for 8 turbine case.

Algo.	LF calls	HF calls	Computational cost (in c_{HF})[b]	AEP (in MWh)[a]	% AEP gain[a]
SCOUT-Nd	—	$600 \cdot 32 \cdot 18$[c]	$345,600c_{HF}$ (1.83x)	$116,472.50$	8.26%
MF-SCOUT-Nd	$615 \cdot 32 \cdot 6$[c]	$615 \cdot 8 \cdot 18$[c]	$\mathbf{127,526.4c_{HF}}$ (0.67x)	$116,214.37$	8.02%
SLSQP	—	$10,484 \cdot 18$	$188,712c_{HF}$ (1x)	$116,978.31$	8.73%

[a] The initial gird layout AEP = 107,581.42 MWh.

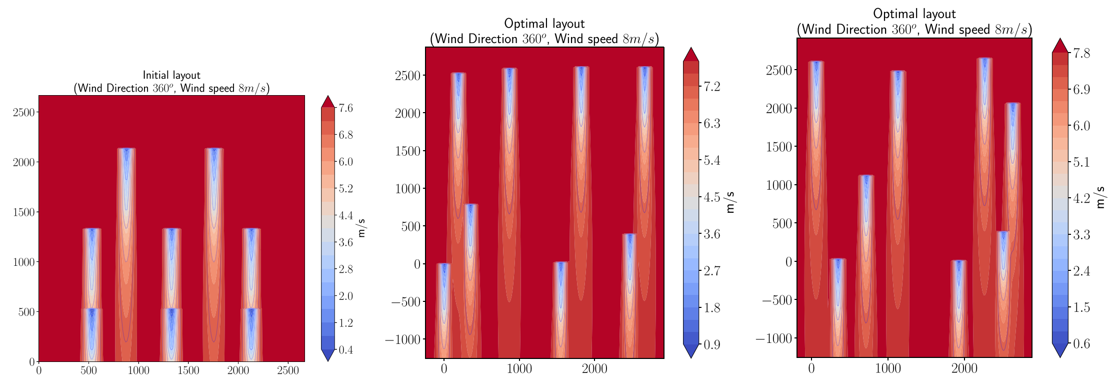
[b] cHF is the cost of a HF run for a single pair of wind direction and wind speed. cLF≈0.33cHF.

[c] Total no. of steps × no. of samples/step × no. of solver call/AEP evaluation.



(a) SLSQP (b) SCOUT-Nd (c) MF-SCOUT-Nd

Figure 14. Windfarm layout with 8 turbines comparing the layout used to initialize (in blue) the optimization routine and the optimized layout (in red) found using SLSQP (a), SCOUT-Nd (b) and MF-SCOUT-Nd (c). The SLSQP and SCOUT-Nd use the GCH model [94], while the MF-SCOUT-Nd uses a combination of the less accurately resolved Jensen wake model [93] and the GCH model. Each dot represents a wind turbine. The blue line represents the boundary of the wind farm. Quantitative Results are given in table 4 and the hyperparameters used to perform the optimization are detailed in table 6.



(a) Initial Layout (b) SCOUT-Nd (c) MF-SCOUT-Nd

Figure 15. Velocity deficit contour plots along the primary wind direction for the initial layout (a), optimized layout with SCOUT-Nd (b), and optimized layout with MF-SCOUT-Nd (c). The corresponding AEPs are given in table 4.

the Latin hypercube as presented in figure 24. With the increase in dimensions, owing to the multimodel nature of the problem, the usual gradient-based optimizers can face this issue, also reported in [10, 15, 88]. This is also reflected in the AEP values reported in the table 5. The SCOUT-Nd reports the highest AEP percentage gain from the initial grid layout, followed by MF-SCOUT-Nd and then the SLSQP. Although the MF-SCOUT-Nd resulted in slightly lower AEP gain as compared to the SCOUT-Nd, interestingly, it took around 1/3 of the cost of SCOUT-Nd optimization with HF. This underscores the capability of our method to be robust and to handle non-convexity in high dimensions (as asserted in section 2.6) in real-world physical applications. By introducing Multi-fidelity, our method allows the analyst to balance the trade-off between the computational cost and the optimum quality.

- The figures 16, 17, 21 and 22 point towards the convergence of SCOUT-Nd and MF-SCOUT-Nd for both the cases by illustrating the evolution of the augmented objective, objective and constraints. As can be seen from the figures, initially the constraints on average are violated. As the optimization progresses, the penalty is also more strongly enforced, thus satisfying the constraints eventually. Further diagnostics of the optimization, like the evolution of the design variables and penalty parameters, are given in appendix F. In all the cases presented, the optimization is terminated by the ϵ_σ convergence criterion for the outer loop. The optimum quality can be further increased by reducing its value, although with an increased computational cost. The

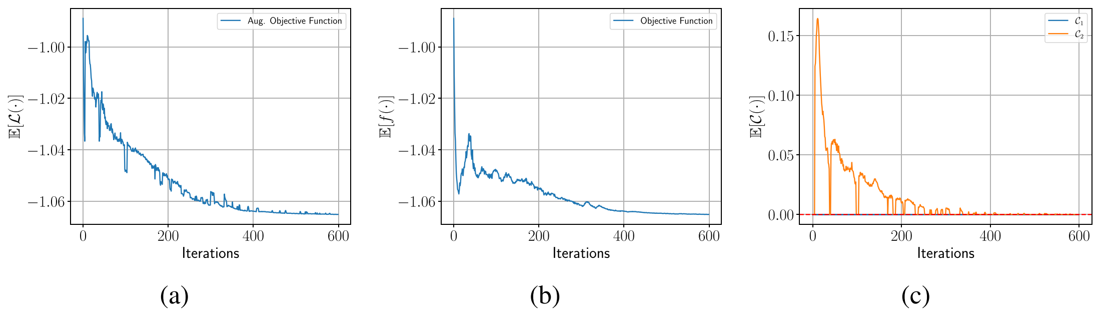


Figure 16. Diagnostics of the optimization run with SCOUT-Nd involving the high-fidelity GCH model [94], for the 8 turbine wind farm case. From left to right, the expected value of augmented objective (a), objective (b), and constraints (c). The objective, constraints, and design variables are normalized. The physical meanings are given in table 3.

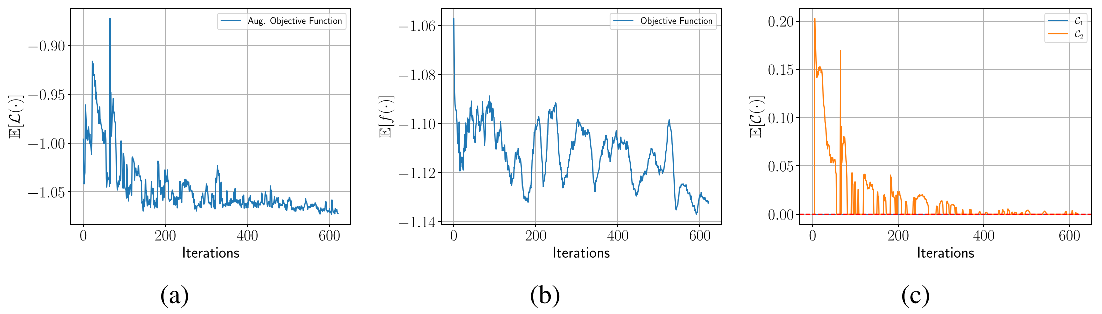


Figure 17. Diagnostics of the optimization run with MF-SCOUT-Nd involving the low-fidelity Jensen wake model [93] and the high-fidelity GCH model [94], for the 8 turbine wind farm case. From left to right, the expected value of augmented objective (a), objective (b), and constraints (c). The objective, constraints, and design variables are normalized. The physical meanings are given in table 3.

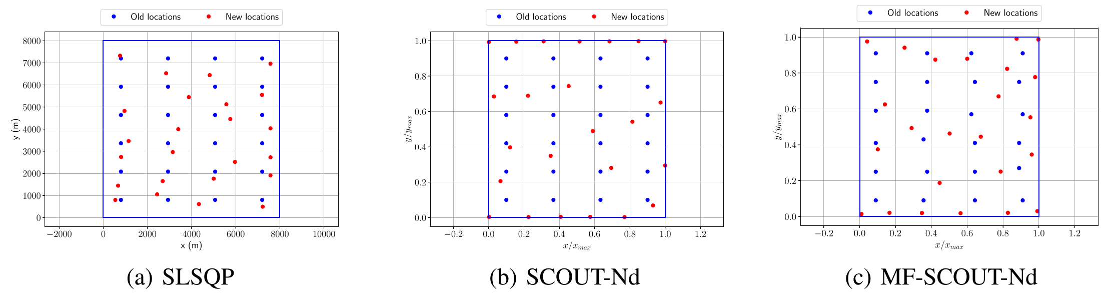


Figure 18. Windfarm layout with 24 turbines comparing the layout used to initialize (in blue) the optimization routine and the optimized layout (in red) found using SLSQP (a), SCOUT-Nd (b) and MF-SCOUT-Nd (c). The SLSQP and SCOUT-Nd use the GCH model [94], while the MF-SCOUT-Nd uses a combination of the less accurately resolved Jensen wake model [93] and the GCH model. Each dot represents a wind turbine. The blue line represents the boundary of the wind farm. Quantitative Results are given in table 5 and the hyperparameters used to perform the optimization are detailed in table 7.

convergence criterion, number of samples, and step size are entwined together in a complicated fashion, influencing the convergence rate and the optimum quality as discussed in sections 2.6.4 and 2.7.1. As with any optimization algorithm [34], we rely on carefully selected heuristics to balance the tradeoff between computational cost and solution accuracy.

4. Conclusions

We presented SCOUT-Nd, a novel approach for (constrained) optimization problems involving expensive, stochastic black-box simulators with high-dimensional parametric dependency. We included strategies and carefully chosen heuristics to handle non-convexity, reduce gradient estimator variance, and improve convergence properties and robustness. We also extended the proposed approach to include MF strategies

Table 5. Optimization results comparison for 24 turbine case.

Algo.	LF calls	HF calls	Computational cost (in c_{HF} [c])	AEP (in MWh) [b]	% AEP gain [b]
SCOUT-Nd	—	$2,648 \cdot 128 \cdot 18$[d]	$6,100,992c_{HF}$ (3.13x)	**347,112.88**	3.98%
MF-SCOUT-Nd	$2,350 \cdot 128 \cdot 6$[d]	$2,350 \cdot 32 \cdot 18$[d]	**$1,949,184c_{HF}$** (1x)	343,662.69	2.98%
SLSQP	—	$7,200 \cdot 18$	$129,600c_{HF}$[a]	336,221.53	0.71%

[a] The optimization run gets trapped in local minima around iteration 120, hence the low AEP and computational cost.
[b] The initial gird layout AEP = 333,834.84 MWh.
[c] c_{HF} is the cost of a HF run for a single pair of wind direction and wind speed. $c_{LF} \approx 0.33c_{HF}$.
[d] Total no. of steps × no. of samples/step × no. of solver call/AEP evaluation.

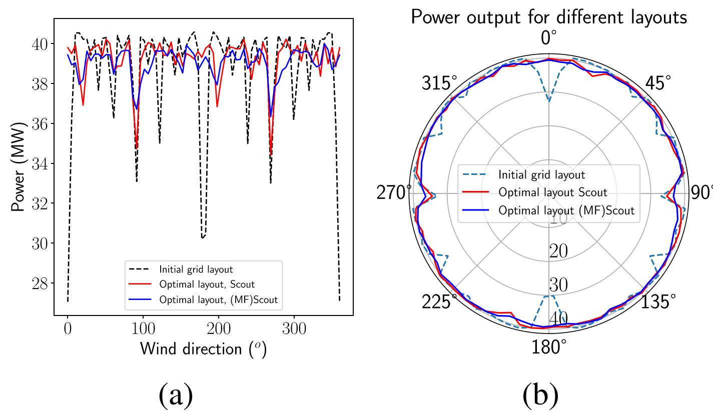


Figure 19. Wind farm power as a function of wind direction for the initial grid layout, SCOUT-Nd optimized layout, and the MF-SCOUT-Nd optimized layout for the 24 turbine case.

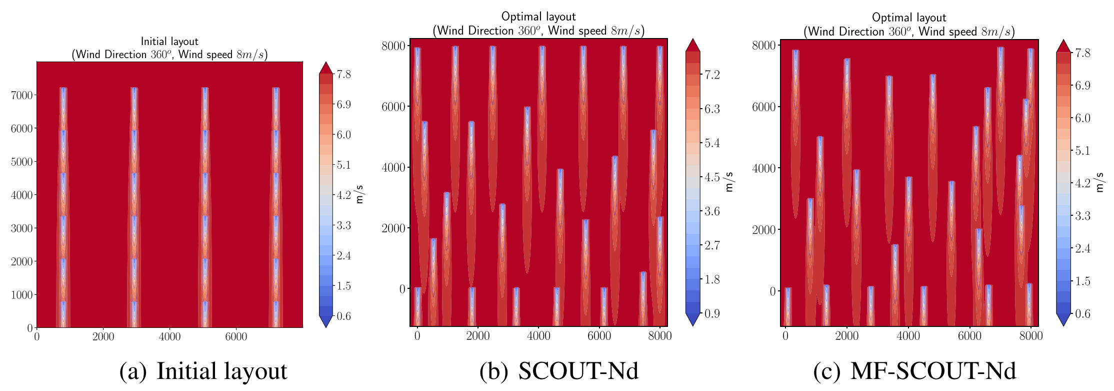


Figure 20. Velocity deficit contour plots along the primary wind direction (from the North side) for the initial layout (a), optimized layout with SCOUT-Nd (b), and optimized layout with MF-SCOUT-Nd (c). The corresponding AEPs are given in table 5.

and adaptive sample size for gradient estimation to limit the number of expensive to evaluate simulator calls. Our method is embarrassingly parallelizable and non-intrusive, thus very attractive to various engineering and physics optimization problems.

We performed experiments on various academic toy problems with common optimization challenges like multi-modality, constraints, behavior in valleys, dimension scalability, etc and compared our algorithm against several baselines. For a given level of accuracy and computational budget, our method solves most of the problems from the toy problem pool. We also demonstrated that it could handle multi-modalities and high-dimensionality better than the baselines. Additionally, we demonstrated the improved quality of the optimum and better convergence rate when including natural gradients. We also tested our algorithm on a complex real-world case of optimizing wind farm layout and compared it against a baseline. We observed that our method improved upon the baseline optimization results, thus showing the successful optimization of a complex stochastic system with a user-defined objective function. Also, the MF-SCOUT-Nd produced comparable accuracy at a one-third cost.

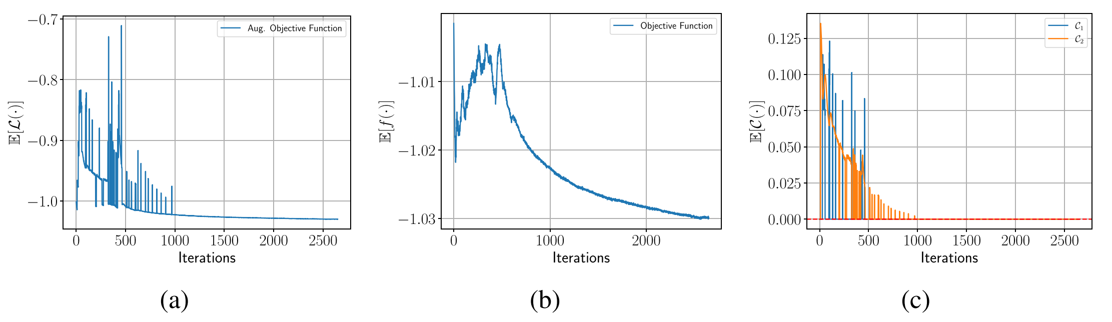


Figure 21. Diagnostics of the optimization run with SCOUT-Nd involving the high-fidelity GCH model [94], for the 24 turbine wind farm case. From left to right, the expected value of augmented objective (a), objective (b), and constraints (c). The objective, constraints, and design variables are normalized. The physical meanings are given in table 3.

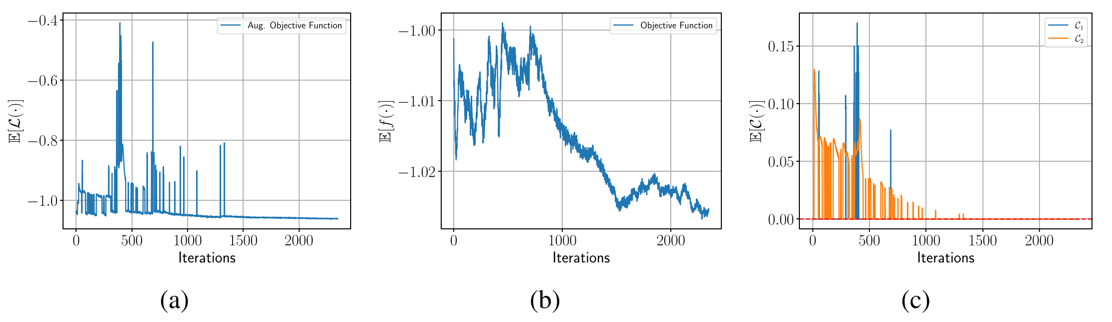


Figure 22. Diagnostics of the optimization run with MF-SCOUT-Nd involving the low-fidelity Jensen wake model [93] and the high-fidelity GCH model [94], for the 24 turbine wind farm case. From left to right, the expected value of augmented objective (a), objective (b), and constraints (c). The objective, constraints, and design variables are normalized. The physical meanings are given in table 3.

The work presented serves as a fertile ground for several extensions and applications. In future research, we intend to adapt and apply the proposed method to optimize hyperparameters in neural networks. Since the algorithm addresses the differentiability issue in problems involving physics-based simulators, integrating into several Scientific Machine Learning paradigms is also possible [25, 96]. This includes hybrid approaches that combine 'known physics' (physics-based simulators) and 'learned physics' (neural network, for example), thus relying on efficient gradient flow. As suggested in [34], we plan to incorporate a full covariance matrix into the design variable density to enhance performance. Furthermore, integrating Importance sampling step into gradient estimation could yield improvements, particularly in scenarios where the objective function landscape contains valleys.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgment

A A and P S K acknowledge the support of VDI Technologiezentrum Gmbh and the Federal Ministry for Education and Research (BMBF) in the context of the project 'Probabilistic ML for the calibration and validation of physics-based models of concrete' (FKZ-13XP5125B). K R and H J B acknowledge the support of Helmholtz Association under the research school 'Munich School for Data Science-MuDs' (HIDSS-006). The authors declare that there is no conflict of interest regarding the publication of this paper.

Appendix A. Analytical expression for the fisher information matrix

Consider the multivariate normal distribution with diagonal covariance, where each variance is parameterized as $\sigma_i^2 = e^{2\beta_i}$.

The log-likelihood function for the multivariate normal distribution is:

$$\log p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \tag{43}$$

Here, the covariance matrix $\boldsymbol{\Sigma}$ is diagonal:

$$\boldsymbol{\Sigma} = \operatorname{diag}\left(e^{2\beta_1}, e^{2\beta_2}, \dots, e^{2\beta_d}\right) \tag{44}$$

Thus, we have:

$$\boldsymbol{\Sigma}^{-1} = \operatorname{diag}\left(e^{-2\beta_1}, e^{-2\beta_2}, \dots, e^{-2\beta_d}\right) \tag{45}$$

Substituting the expressions for $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}^{-1}$ in equation (43):

$$\log p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\beta}) = -\frac{d}{2}\log(2\pi) - \sum_{i=1}^{d}\beta_i - \frac{1}{2}\sum_{i=1}^{d}\frac{(x_i-\mu_i)^2}{e^{2\beta_i}} \tag{46}$$

The Fisher Information for $\boldsymbol{\mu}$ comes from the derivative of the log-likelihood with respect to μ_i. Taking the derivative:

$$\frac{\partial \log p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \mu_i} = \frac{x_i-\mu_i}{e^{2\beta_i}} \tag{47}$$

Since $\mathbb{E}[(x_i-\mu_i)^2] = e^{2\beta_i}$, the Fisher Information for μ_i is then:

$$F(\mu_i) = \mathbb{E}\left[\left(\frac{x_i-\mu_i}{e^{2\beta_i}}\right)^2\right] = \frac{1}{e^{2\beta_i}} \tag{48}$$

Thus, the FIM for $\boldsymbol{\mu}$ is diagonal:

$$F(\boldsymbol{\mu}) = \operatorname{diag}\left(\frac{1}{e^{2\beta_1}}, \frac{1}{e^{2\beta_2}}, \dots, \frac{1}{e^{2\beta_d}}\right) \tag{49}$$

Now, we calculate the Fisher Information for β_i, given that $\sigma_i^2 = e^{2\beta_i}$. The derivative of the log-likelihood with respect to β_i is:

$$\frac{\partial \log p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\beta})}{\partial \beta_i} = -1 + \frac{(x_i-\mu_i)^2}{e^{2\beta_i}} \tag{50}$$

The Fisher Information is given by the expected value of the square of the derivative:

$$F(\beta_i) = \mathbb{E}\left[\left(-1 + \frac{(x_i-\mu_i)^2}{e^{2\beta_i}}\right)^2\right] \tag{51}$$

$$= 1 - 2\mathbb{E}\left[\frac{(x_i-\mu_i)^2}{e^{2\beta_i}}\right] + \mathbb{E}\left[\frac{(x_i-\mu_i)^4}{e^{4\beta_i}}\right] \tag{52}$$

Since $\mathbb{E}[(x_i-\mu_i)^2] = e^{2\beta_i}$ and $\mathbb{E}[(x_i-\mu_i)^4] = 3e^{4\beta_i}$ (fourth central moment), the expression simplifies to:

$$F(\beta_i) = 1 - 2\frac{e^{2\beta_i}}{e^{2\beta_i}} + 3\frac{3e^{4\beta_i}}{e^{4\beta_i}} \tag{53}$$

$$= 1 - 2 + 3 = 2 \tag{54}$$

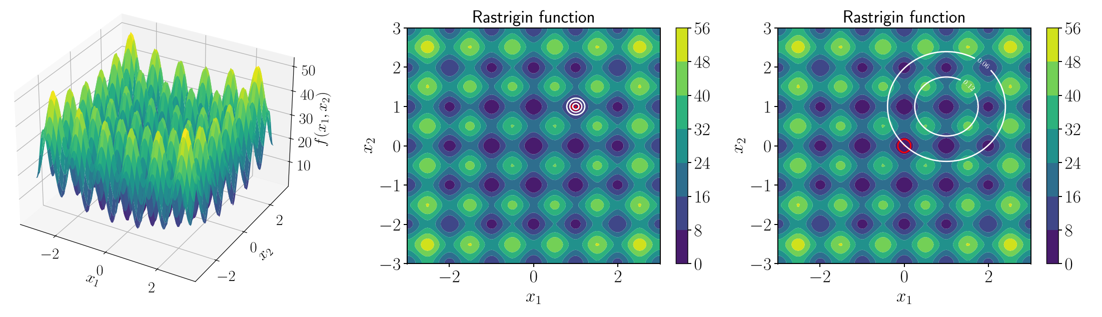
Thus the FIM for $\boldsymbol{\beta}$ is:

$$F(\boldsymbol{\beta}) = 2\boldsymbol{I}_d \tag{55}$$

The FIM for both $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ is block-diagonal. Combining the results, we get:

$$\boldsymbol{F}(\boldsymbol{\mu}, \boldsymbol{\beta}) = \operatorname{diag}\left(e^{-2\beta_1}, \dots, e^{-2\beta_d}, 2\boldsymbol{I}_d\right) \tag{56}$$

Appendix B. Influence of initial value of design variable variance



(a) Rastringin function surface (b) Initial value of $\mathbf{\Sigma} = e^{-2}\boldsymbol{I}$ (c) Initial value of $\mathbf{\Sigma} = e^{0}\boldsymbol{I}$

Figure 23. Using 2d Rastringin function to highlight the influence of the initial value of design variance on escaping local optima. White contour lines represent the initial $q(\boldsymbol{\theta})$ and red contour lines represent the final design variable distribution $q(\boldsymbol{\theta}^*)$.

Appendix C. Taylor expansion simplification

Our proposed algorithm relies on optimizing $U(\boldsymbol{\theta})$ instead of $f(\boldsymbol{x})$. Let us convert the distribution to standard normal distribution as follows:

$$U(\boldsymbol{\theta}) = \mathbb{E}\left[f(\boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{z})\right]_{\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})}. \tag{57}$$

Writing down the Taylor expansion of $f(\boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{z})$ till the third derivative:

$$f(\boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{z}) = f(\boldsymbol{\mu}) + \sum_{i=1}^{d} \sigma_i z_i \frac{\partial f}{\partial x_i}(\boldsymbol{\mu}) + \frac{1}{2!} \sum_{i,j=1}^{d} \sigma_i \sigma_j z_i z_j \frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{\mu}) \dots$$
$$+ \frac{1}{3!} \sum_{i,j,k=1}^{d} \sigma_i \sigma_j \sigma_k z_i z_j z_k \frac{\partial^3 f}{\partial x_i \partial x_j \partial x_k}(\boldsymbol{\mu}) + O\left(\|\boldsymbol{\sigma}\|^4\right),$$

where z_i, σ_i and x_i are the ith component of $\boldsymbol{z}$, $\boldsymbol{\sigma}$ and $\boldsymbol{x}$ respectively. We take expectation on both sides to get $U(\boldsymbol{\theta})$.

$$\begin{aligned} U(\boldsymbol{\theta}) &= \mathbb{E}\left[f(\boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{z})\right]_{\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \\ &= f(\boldsymbol{\mu}) + \sum_{i=1}^{d} \sigma_i \mathbb{E}\left[z_i\right] \frac{\partial f}{\partial x_i}(\boldsymbol{\mu}) + \frac{1}{2!} \sum_{i,j=1}^{d} \sigma_i \sigma_j \mathbb{E}\left[z_i z_j\right] \frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{\mu}) \dots \\ &\quad + \frac{1}{3!} \sum_{i,j,k=1}^{d} \sigma_i \sigma_j \sigma_k \mathbb{E}\left[z_i z_j z_k\right] \frac{\partial^3 f}{\partial x_i \partial x_j \partial x_k}(\boldsymbol{\mu}) + O\left(\|\boldsymbol{\sigma}\|^4\right). \end{aligned} \tag{58}$$

We simplify the above expression using the following observations:

$$\mathbb{E}\left[z_i\right] = 0 \quad \text{because of } 0 \text{ mean}$$

$$\mathbb{E}\left[z_i z_j\right] = \begin{cases} 1 & \text{if } i = j \text{ because standard deviation of the distribution is } 1 \\ 0 & \text{otherwise because the variables are independent and have zero mean} \end{cases}$$

$$\mathbb{E}\left[z_i z_j z_k\right] = 0 \quad \text{because skewness of normal distribution is zero}$$

The simplified expression is:

$$U(\boldsymbol{\theta}) = f(\boldsymbol{\mu}) + \frac{1}{2} \sum_{i=1}^{d} \sigma_i^2 \frac{\partial^2 f}{\partial x_i^2}(\boldsymbol{\mu}) + O\left(\|\boldsymbol{\sigma}\|^4\right) \tag{59}$$

Appendix D. List of benchmarks

We represent the dimension of the search space by d_p. The list of benchmarks is as follows:

D.1. Sphere problem

We run this problem for dimensions $d_p = \{2,4,8,16,32\}$. The HF and the LF function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= \sum_{i=1}^{d} x_i^2, \\ f_{\text{low}}(x) &= \sum_{i=1}^{d} 1.1 x_i^2. \end{aligned} \tag{60}$$

It is a simple problem with a single global minimum at the $(0,\dots d_p$ times).

D.2. Constrained sphere problem

We run this problem for dimensions $d_p = \{2,4,8,16,32\}$. The objection function is same as equation (60) but with a constraint $1 - x_1 - x_2 \leqslant 0$. The optimum lies on the constraint boundary at $(0.5, 0.5, 0, \dots d_p - 2\text{times})$. The problem checks the ability of the algorithm to handle constraints.

D.3. Ackley function

We run this problem for dimensions $d_p = \{2,4,8,16,32\}$. The HF and the LF function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= -20\exp\left(-0.2\sqrt{\frac{1}{d_p}\sum_{i=1}^{d_p} x_i^2}\right) - \exp\left(\frac{1}{d_p}\sum_{i=1}^{d_p}\cos(2\pi x_i)\right) + 20 + \exp(1), \\ f_{\text{low}}(x) &= -22\exp\left(-0.2\sqrt{\frac{1}{d_p}\sum_{i=1}^{d_p} 1.1x_i^2}\right) - 0.9\exp\left(\frac{1}{d_p}\sum_{i=1}^{d_p}\cos(2\pi x_i)\right) + 20 + \exp(1). \end{aligned} \tag{61}$$

It is a multimodal problem with a global minimum at $(0,\dots d\text{times})$.

D.4. Rosenbrock function

We run this problem for dimensions $d_p = \{2,4,8,16\}$. The HF and the LF function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= \sum_{i=1}^{d_p-1} 100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2, \\ f_{\text{low}}(x) &= \sum_{i=1}^{d_p-1} 101\left(x_{i+1} - x_i^2\right)^2 + 1.01(1 - x_i)^2 + 0.2. \end{aligned} \tag{62}$$

The function is unimodal, but the minimum lies in a narrow valley is at $(1,1,\dots d_p\text{times})$. The optimizers find the valley of the problem but take a long time to converge to the minimum [85].

D.5. Zakharov function

We run this problem for dimensions $d_p = \{2,4,8,16\}$. The HF and the LF function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= \sum_{i=1}^{d_p} x_i^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{d_p} 0.5ix_i\right)^4, \\ f_{\text{low}}(x) &= \sum_{i=1}^{d_p} x_i^2 + \left(\sum_{i=1}^{d} 0.55ix_i\right)^2 + \left(\sum_{i=1}^{d_p} 0.5ix_i\right)^4. \end{aligned} \tag{63}$$

It has one minimum at $(0,\dots d_p\text{times})$. It is a plate-shaped function. Optimizers tend to get stuck in the flat regions of the function.

D.6. Bohachevsky function: 1

This is a 2-dimensional bowl-shaped problem. The HF and the LF function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7, \\ f_{\text{low}}(x) &= x_1^2 + 2.1x_2^2 - 0.32\cos(3\pi x_1) - 0.41\cos(4\pi x_2) + 0.7. \end{aligned} \tag{64}$$

The minimum lies at $(0, 0)$.

D.7. Bohachevsky function: 2

This is a 2-dimensional bowl-shaped problem. The HF and the LF function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3, \\ f_{\text{low}}(x) &= x_1^2 + 2.1x_2^2 - 0.31\cos(3\pi x_1)\cos(4\pi x_2) + 0.3. \end{aligned} \tag{65}$$

The minima lies at $(0, 0)$.

D.8. Six-hump camel function

This is a 2-dimensional function with six local minima. The HF and the LF function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= 4x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3} + x_1x_2 - 4x_2^2 + 4x_2^4, \\ f_{\text{low}}(x) &= 4x_1^2 - 2.2x_1^4 + \frac{x_1^6}{3.2} + 1.1x_1x_2 - 4.1x_2^2 + 4x_2^4. \end{aligned} \tag{66}$$

There are two global minima at $(0.0898, -0.7126)$ and $(0.0898, 0.7126)$.

D.9. Three-hump camel function

This is a 2-dimensional function with three local minima. The HF and the LF function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2, \\ f_{\text{low}}(x) &= 2.1x_1^2 - 1.06x_1^4 + \frac{x_1^6}{6} + 1.1x_1x_2 + x_2^2. \end{aligned} \tag{67}$$

D.10. Beale function

This is a 2-dimensional multimodal problem. The HF and the LF function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= (1.5 - x_1 + x_1x_2)^2 + \left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x_1 + x_1x_2^3\right)^2, \\ f_{\text{low}}(x) &= (1.54 - x_1 + x_1x_2)^2 + \left(2.29 - x_1 + x_1x_2^2\right)^2 + \left(2.675 - x_1 + x_1x_2^3\right)^2. \end{aligned} \tag{68}$$

The minima lies at $(3, 0.5)$.

D.11. Hartmann 3d function

This is a 3-dimensional problem with four local minima. The HF and the LF function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}\left(x_j - P_{ij}\right)^2\right), \\ f_{\text{low}}(x) &= -1.1\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}\left(x_j - P_{ij}\right)^2\right) + 0.1, \end{aligned} \tag{69}$$

where α, A and P are defined as:

$$\alpha = (1, 1.2, 3, 3.2)^T$$

$$A = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}$$

$$P = \begin{pmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.0381 & 0.5743 & 0.8828 \end{pmatrix}$$

The global minima lies at $(0.114614, 0.555649, 0.852547)$.

D.12. Hartmann 4d function

This is a multimodal 4-dimensional problem. The HF and the LF function are given by:

$$\begin{aligned} f_{\text{high}}(x) &= -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{4} A_{ij}\left(x_j - P_{ij}\right)^2\right), \\ f_{\text{low}}(x) &= -1.1\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{4} A_{ij}\left(x_j - P_{ij}\right)^2\right) + 0.1, \end{aligned} \tag{70}$$

where α, A and P are defined as:

$$\alpha = (1, 1.2, 3, 3.2)^T$$

$$A = \begin{pmatrix} 10 & 3 & 17 & 3.5 \\ 0.05 & 10 & 17 & 0.1 \\ 3 & 3.5 & 1.7 & 10 \\ 17 & 8 & 0.05 & 10 \end{pmatrix}$$

$$P = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 \end{pmatrix}$$

The global minima lies at $(0.1873, 0.1906, 0.5566, 0.2647)$.

Appendix E. Hyperparameters for benchmark studies

Hyper-parameters of SCOUT-Nd and MF-SCOUT-nd are

1. $\eta_{step} = 0.1$, $\sigma_{\text{cut-off}} = 0.0005$
2. Number of samples is in table 2
3. We run the optimizer for 2000 steps.
4. For constrained problems: We use five penalty constants $(\{e^{-1}, e^0, e^1, e^2, e^3\})$. We run the optimizer on each penalty term for maximum of 400 steps.

Hyper-parameters of SLSQP:

1. Function value tolerance(`ftol`) is 10^{-12}.
2. Maximum number of iterations is 2000.
3. Step size used for numerical approximation of the Jacobian(`eps`) is 0.001.

Hyper-parameters of COBYLA

1. Function value tolerance(`tol`) is 10^{-10}.
2. Maximum number of iterations is 2000.

Hyper-parameters of cBO

1. Acquisition function: Expected Improvement.
2. We use Materm Kernel with $\nu = 4/5$.
3. The hyper-parameters of the kernel is optimized by ARD, with 6 re-starts.
4. Maximum number of iterations is 2000.

Hyper-parameters of DYCOR

1. Symmetric Latin hypercube experimental design.
2. Radial basic function.
3. Cubic kernel function and Linear tail.
4. Maximum number of optimization steps is 2000.
5. Cut-off standard deviation is 0.0001.

Appendix F. Further details of the windfarm layout optimization

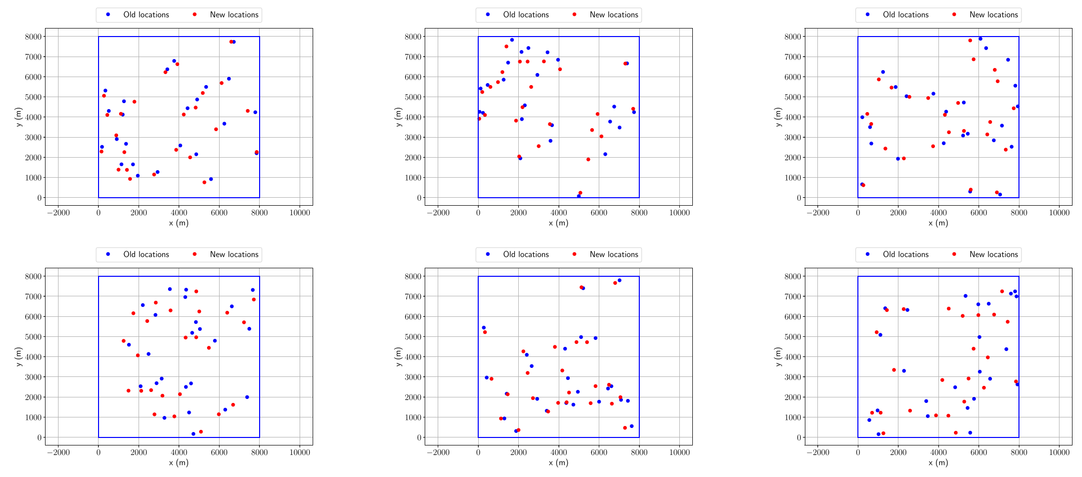


Figure 24. Windfarm layout optimization with SLSQP for 24 turbine case with 6 different initial layouts sampled with latin hypercube. In the six runs, the final layout never reaches the boundaries (as expected for an optimal layout) and the optimizer gets trapped in the local optima.

In the windfarm layout optimization case, we observed that for high dimensional optimization, the SLSQP was trapped in local optima when optimization was started from a grid layout. To confirm this behavior, we ran the SLSQP optimizer using different initial layouts drawn from the latin hypercube as presented in figure 24.

Tables 6 and 7 show the hyperparameters of the algorithms used for the windfarm layout optimization case for 8 turbine and 24 turbine cases, respectively.

The figures 25–28 provides optimization diagnostics of SCOUT-Nd and MF-SCOUT-Nd for both the cases. In all four figures, it is interesting to observe the normalized design variable mean fluctuation near the wind farm boundary around the initial iterations and the corresponding constraint violation. Gradually as the penalty is more strongly enforced (by increasing the penalty parameter λ) the constraints are satisfied on average, accompanied by stabilization of the design variable mean.

Table 6. Hyper-parameters of algorithms used to carry out experiments for 8 turbine windfarm case.

SCOUT-Nd		MF-SCOUT-Nd		SLSQP [a]	
parameter	value	parameter	value	parameter	value
η_{step}	$5e-02$	η_{step}	$5e-02$	`ftol`	$1e-10$
S_{HF}	32	(S_{LF}, S_{HF})	(32,8)	`maxiter`	1200
N_{max}^{inner}	300	N_{max}^{inner}	300	`eps`	0.01
N_{max}^{outer}	1200	N_{max}^{outer}	1200	—	—
$\sigma_{\text{cut-off}}$	$1e-04$	$\sigma_{\text{cut-off}}$	$1e-04$	—	—
$\{\boldsymbol{\lambda}_m\}$	e^{-1}, e^0	$\{\boldsymbol{\lambda}_m\}$	e^{-1}, e^0, e^1	—	—

[a] The parameters of SLSQP correspond to the parameters in its Scipy implementation. More details can be found https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html here and [50].

Table 7. Hyper-parameters of algorithms used to carry out experiments for 24 turbine windfarm case.

SCOUT-Nd		MF-SCOUT-Nd		SLSQP	
parameter	value	parameter	value	parameter	value
η_{step}	$1e-02$	η_{step}	$1e-02$	`ftol`	$1e-12$
S	128	(S_{LF}, S_{HF})	(12 832)	`maxiter`	3000
N_{max}^{inner}	300	N_{max}^{inner}	300	`eps`	0.01
N_{max}^{outer}	3000	N_{max}^{outer}	3000	—	—
$\sigma_{\text{cut-off}}$	$5e-04$	$\sigma_{\text{cut-off}}$	$5e-04$	—	—
$\{\boldsymbol{\lambda}_m\}$	$\{e^{-1}, e^0, \ldots, e^8\}$	$\{\boldsymbol{\lambda}_m\}$	$\{e^{-1}, e^0, \ldots, e^7\}$	—	—

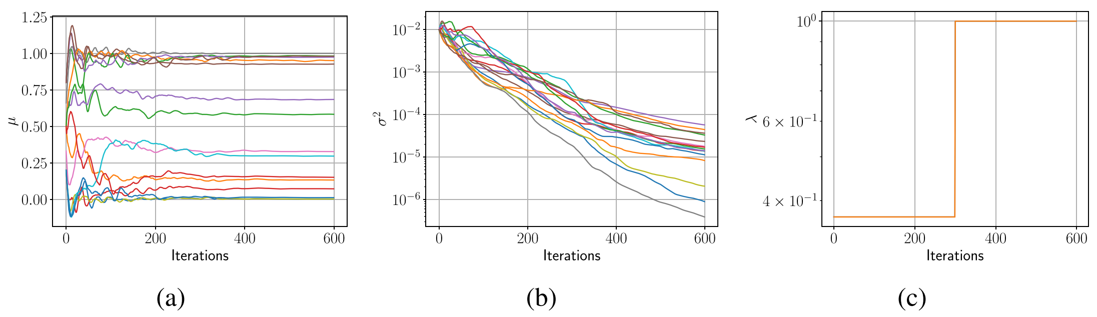


Figure 25. Diagnostics of the optimization run with SCOUT-Nd involving the high-fidelity GCH model [94], for the 8 turbine wind farm case. From left to right, the mean of the design variables (a), the variance of the design variables (b), and the penalty term λ (c).

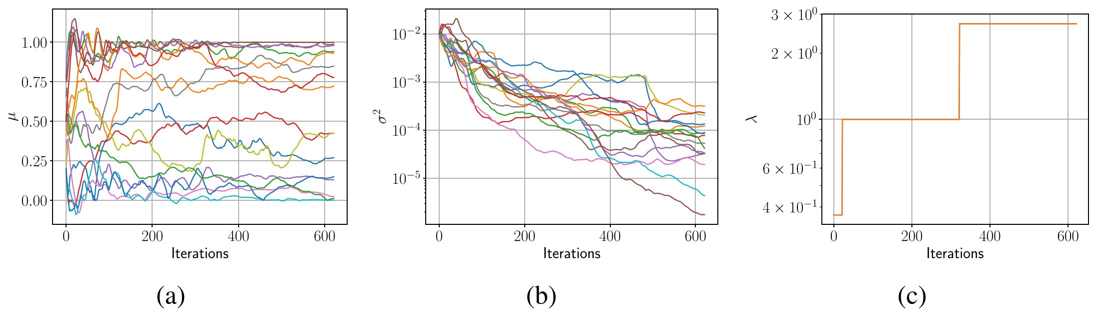


Figure 26. Diagnostics of the optimization run with MF-SCOUT-Nd involving the high-fidelity GCH model [94], for the 8 turbine wind farm case. From left to right, the mean of the design variables (a), the variance of the design variables (b), and the penalty term λ (c).

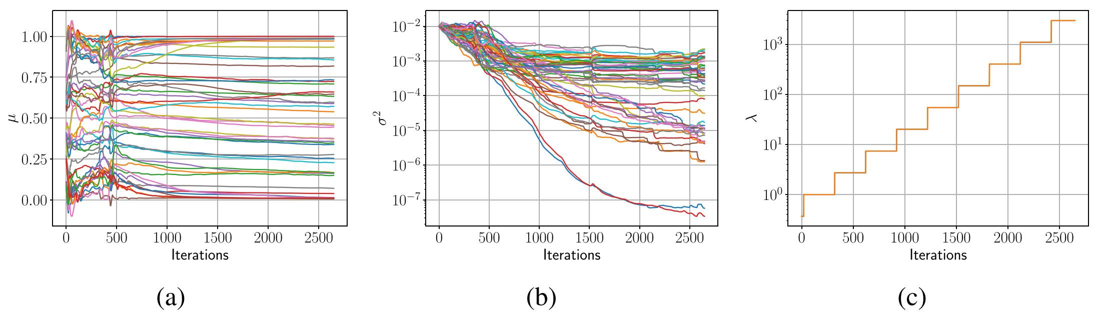


Figure 27. Diagnostics of the optimization run with SCOUT-Nd involving the high-fidelity GCH model [94], for the 24 turbine wind farm case. From left to right, the mean of the design variables (a), the variance of the design variables (b), and the penalty term λ (c).

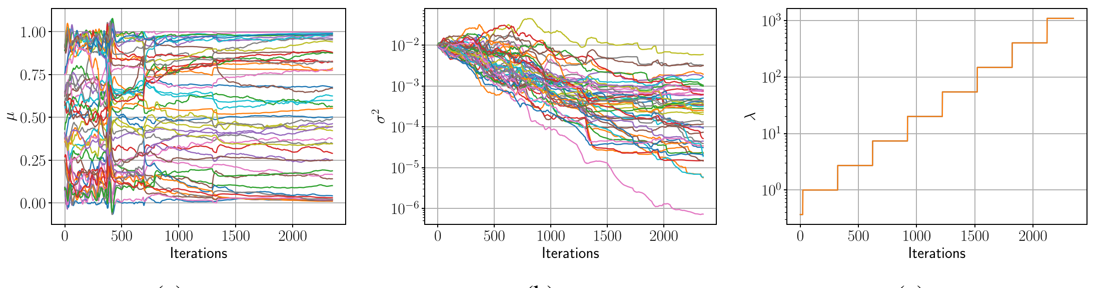


Figure 28. Diagnostics of the optimization run with MF-SCOUT-Nd involving the high-fidelity GCH model [94], for the 24 turbine wind farm case. From left to right, the mean of the design variables (a), the variance of the design variables (b), and the penalty term λ (c).

ORCID iDs

Atul Agrawal https://orcid.org/0000-0002-9101-359X
Kislaya Ravi https://orcid.org/0000-0003-1927-7651
Phaedon-Stelios Koutsourelakis https://orcid.org/0000-0002-9345-759X
Hans-Joachim Bungartz https://orcid.org/0000-0002-0171-0712

References

[1] Cranmer K, Brehmer J and Louppe G 2020 The frontier of simulation-based inference *Proc. of the National Academy of Sciences Proc. Natl Acad. Sci.* **117** 30055–62
[2] Reuther J, Jameson A, Farmer J, Martinelli L and Saunders D 1996 Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation *34th Aerospace Sciences Meeting and Exhibit* p 94
[3] Hasenjäger M, Sendhoff B, Sonoda T and Arima T 2005 Three dimensional evolutionary aerodynamic design optimization with cma-es *Proc. 7th Annual Conf. on Genetic and Evolutionary Computation* pp 2173–80
[4] Jebalia M, Auger A, Schoenauer M, James F and Postel M 2007 Identification of the isotherm function in chromatography using CMA-ES *2007 IEEE Congress on Evolutionary Computation* (IEEE) pp 4289–96
[5] Agrawal A and Koutsourelakis P-S 2024 A probabilistic, data-driven closure model for rans simulations with aleatoric, model uncertainty *J. Comput. Phys.* **08** 112982
[6] Agrawal A, Tamsen E, Koutsourelakis P-S, and Unger J F 2023 From concrete mixture to structural design–a holistic optimization procedure in the presence of uncertainties (arXiv:2312.03607)
[7] Rixner M and Koutsourelakis P-S 2022 Self-supervised optimization of random material microstructures in the small-data regime *npj Comput. Mater.* **8** 46
[8] Shirobokov S, Belavin V, Kagan M, Ustyuzhanin A and Baydin A G 2020 Black-box optimization with local generative surrogates *Advances in Neural Information Processing Systems* vol 33 pp 14650–62
[9] Dorigo T *et al* 2023 Toward the end-to-end optimization of particle physics instruments with differentiable programming *Rev. Phys.* **10** 100085
[10] Thomas J J, Baker N F, Malisani P, Quaeghebeur E, Perez-Moreno S S, Jasa J, Bay C, Tilli F, Bieniek D and Robinson N 2022 A comparison of eight optimization methods applied to a wind farm layout optimization problem *Wind Energy Sci. Discuss.* **2022** 1–43
[11] Zhang Z, Song M and Huang X 2020 Online accelerator optimization with a machine learning-based stochastic algorithm *Mach. Learn.: Sci. Technol.* **2** 015014
[12] Williams R J 1992 Simple statistical gradient-following algorithms for connectionist reinforcement learning *Mach. Learn.* **8** 229–56
[13] Ruiz N, Schulter S, and Chandraker M 2018 Learning to simulate (arXiv:1810.02513)
[14] Ros G, Sellart L, Materzynska J, Vazquez D and Lopez A M 2016 The synthia dataset: a large collection of synthetic images for semantic segmentation of urban scenes *Proc. IEEE Conf. onComputer Vision and Pattern Recognition* pp 3234–43
[15] Martins J R and Ning A 2021 *Engineering Design Optimization* (Cambridge University Press)
[16] Blondel M and Roulet V 2024 The elements of differentiable programming (arXiv:2403.14606)
[17] Degrave J, Hermans M and Dambre J 2019 A differentiable physics engine for deep learning in robotics *Front. Neurorobot.* **16** 6
[18] de Avila Belbute-Peres F, Smith K, Allen K, Tenenbaum J and Kolter J Z 2018 End-to-end differentiable physics for learning and control *Advances in Neural Information Processing Systems* vol 31
[19] Lucor D, Agrawal A and Sergent A 2022 Simple computational strategies for more effective physics-informed neural networks modeling of turbulent natural convection *J. Comput. Phys.* **456** 111022
[20] Golovin D, Solnik B, Moitra S, Kochanski G, Karro J and Sculley D 2017 Google vizier: A service for black-box optimization *Proc. 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* pp 1487–95
[21] Giles M B and Pierce N A 2000 An introduction to the adjoint approach to design *Flow, Turbulence combust.* **65** 393–415
[22] Luce A, Alaee R, Knorr F and Marquardt F 2024 Merging automatic differentiation and the adjoint method for photonic inverse design *Mach. Learn.: Sci. Technol.* **5** 025076
[23] Holl P and Thuerey N 2024 Phi-ml: intuitive scientific computing with dimension types for jax, pytorch, tensorflow & numpy *J. Open Source Softw.* **9** 6171
[24] Gasiorowski S, Chen Y, Nashed Y, Granger P, Mironov C, Tsang K V, Ratner D and Terao K 2024 Differentiable simulation of a liquid argon time projection chamber *Mach. Learn.: Sci. Technol.* **5** 025012

[25] Baker N, *et al* 2019 Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence, USDOE Office of Science (SC),Washington, DC (United States) *Technical Report*

[26] Moses W and Churavy V 2020 Instead of rewriting foreign code for machine learning, automatically synthesize fast gradients *Advances in Neural Information Processing Systems* vol 33 12472–85

[27] Larson J, Menickelly M and Wild S M 2019 Derivative-free optimization methods *Acta Numer.* **28** 287–404

[28] Audet C and Kokkolaras M 2016 Blackbox and derivative-free optimization: theory, algorithms and applications *Optim. Eng.* **17** 1–2

[29] Moré J J and Wild S M 2009 Benchmarking derivative-free optimization algorithms *SIAM J. Optim.* **20** 172–91

[30] Ranganath R, Gerrish S and Blei D 2014 Black box variational inference *Artif. Intell. Stat.* **33** 814–22

[31] Banzhaf W, Nordin P, Keller R E and Francone F D 1998 *Genetic Programming: an Introduction: on the Automatic Evolution of Computer Programs and its Applications* (Morgan Kaufmann Publishers Inc)

[32] Snoek J, Larochelle H and Adams R P 2012 Practical bayesian optimization of machine learning algorithms *Advances in Neural Information Processing Systems* vol 25

[33] Menhorn F, Augustin F, Bungartz H-J and Marzouk Y M 2017 A trust-region method for derivative-free nonlinear constrained stochastic optimization (arXiv:1703.04156)

[34] Wierstra D, Schaul T, Glasmachers T, Sun Y, Peters J and Schmidhuber J 2014 Natural evolution strategies *J. Mach. Learn. Res.* **15** 949–80

[35] Choromanski K M, Pacchiano A, Parker-Holder J, Tang Y and Sindhwani V 2019 From complexity to simplicity: adaptive es-active subspaces for blackbox optimization *Advances in Neural Information Processing Systems* vol 32

[36] Anand A, Degroote M and Aspuru-Guzik A 2021 Natural evolutionary strategies for variational quantum computation *Mach. Learn.: Sci. Technol.* **2** 045012

[37] Mohamed S, Rosca M, Figurnov M and Mnih A 2020 Monte carlo gradient estimation in machine learning *J. Mach. Learn. Res.* **21** 5183–244

[38] Pflug G C 2012 *Optimization of Stochastic Models: the Interface Between Simulation and Optimization* vol 373 (Springer)

[39] Louppe G, Hermans J and Cranmer K 2019 Adversarial variational optimization of non-differentiable simulators *Proc. 22nd Int. Conf. on Artificial Intelligence and Statistics* (PMLR) pp 1438–47

[40] Tucker G, Mnih A, Maddison C J, Lawson J and Sohl-Dickstein J 2017 Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models *Advances in Neural Information Processing Systems* vol 30

[41] Grathwohl W, Choi D, Wu Y, Roeder G and Duvenaud D 2018 Backpropagation through the void: Optimizing control variates for black-box gradient estimation *Int. Conf. on Learning Representations*

[42] Agrawal A, Ravi K, Koutsourelakis P-S, and Bungartz H-J 2023 Multi-fidelity constrained optimization for stochastic black box simulators (arXiv:2311.15137)

[43] Nemirovskij A S and Yudin D B 1983 Problem complexity and method efficiency in optimization

[44] Nguyen A and Balasubramanian K 2023 Stochastic zeroth-order functional constrained optimization: Oracle complexity and applications *INFORMS J. Optim.* **5** 256–72

[45] Bird T, Kunze J, and Barber D 2018 Stochastic variational optimization (arXiv:1809.04855 [cs, stat])

[46] Staines J and Barber D 2012 Variational optimization (arXiv:1212.4507 [cs, stat])

[47] Gardner J R, Kusner M J, Xu Z E, Weinberger K Q and Cunningham J P 2014 Bayesian optimization with inequality constraints *ICML* **2014** 937–45

[48] Powell M J 1994 *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation* (Springer)

[49] Regis R G and Shoemaker C A 2013 Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization *Eng. Optim.* **45** 529–55

[50] Kraft D 1988 A software package for sequential quadratic programming, Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt

[51] Bliek L, Guijt A, Karlsson R, Verwer S, and de Weerdt M 2021 Expobench: benchmarking surrogate-based optimisation algorithms on expensive black-box functions (arXiv:2106.04618)

[52] Ben-Tal A and Nemirovski A 1999 Robust solutions of uncertain linear programs *Oper. Res. Lett.* **25** 1–13

[53] Bertsimas D, Brown D B and Caramanis C 2011 Theory and applications of robust optimization *SIAM Rev.* **53** 464–501

[54] Wang I-J and Spall J 2003 Stochastic optimization with inequality constraints using simultaneous perturbations and penalty functions *42nd IEEE Int. Conf. on Decision and Control* (IEEE Cat. No.03CH37475) (*Maui, HI, USA*) (IEEE) pp 3808–13

[55] Nocedal J and Wright S J 1999 *Numerical Optimization* (Springer)

[56] Fiacco A V and McCormick G P 1990 *Nonlinear Programming: Sequential Unconstrained Minimization Techniques* (SIAM)

[57] Liuzzi G, Lucidi S and Sciandrone M 2010 Sequential penalty derivative-free methods for nonlinear constrained optimization *SIAM J. Optim.* **20** 2614–35

[58] Fortin M and Glowinski R 2000 *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems* (Elsevier)

[59] Beaumont M A, Zhang W and Balding D J 2002 Approximate bayesian computation in population genetics *Genetics* **162** 2025–35

[60] Marjoram P, Molitor J, Plagnol V and Tavaré S 2003 Markov chain monte carlo without likelihoods *Proc. Natl Acad. Sci.* **100** 15324–8

[61] Staines J and Barber D 2013 Optimization by variational bounding *European Symp. on Artificial Neural Networks, Computational Intelligence And Machine Learning* (Bruges, *24–26 April 2013*) (available at: https://www.esann.org/sites/default/files/proceedings/legacy/es2013-65.pdf)

[62] Glynn P W 1990 Likelihood ratio gradient estimation for stochastic systems *Commun. ACM* **33** 75–84

[63] Salimans T, Ho J, Chen X, Sidor S, and Sutskever I 2017 Evolution strategies as a scalable alternative to reinforcement learning (arXiv:1703.03864)

[64] Sehnke F, Osendorfer C, Rückstieß T, Graves A, Peters J and Schmidhuber J 2010 Parameter-exploring policy gradients *Neural Netw.* **23** 551–9

[65] Nesterov Y and Spokoiny V 2017 Random gradient-free minimization of convex functions *Found. Comput. Math.* **17** 527–66

[66] Kool W, v. Hoof H, and Welling M 2022 Buy 4 reinforce Samples, Get a Baseline for Free! (available at: https://openreview.net/forum?id= r1lgTGL5DE)

[67] Dick J, Kuo F Y and Sloan I H 2013 High-dimensional integration: the quasi-monte carlo way *Acta Numer.* **22** 133–288

[68] Rowland M, Choromanski K M, Chalus F, Pacchiano A, Sarlos T, Turner R E and Weller A 2018 Geometrically coupled monte carlo sampling *Advances in Neural Information Processing Systems* vol 31

[69] Glasserman P 2004 *Monte Carlo Methods in Financial Engineering* (Springer) p 53
[70] Sobol' I M 1967 On the distribution of points in a cube and the approximate evaluation of integrals *Z. Vychisl. Mat. Mat. Fiz.* **7** 784–802
[71] Amari S-I 1998 Natural gradient works efficiently in learning *Neural Comput.* **10** 251–76
[72] Tang D and Ranganath R 2019 The variational predictive natural gradient *Proc. 36th Int. Conf. on Machine Learning* (*Proc. Machine Learning Research*) vol 97 ed K Chaudhuri and R Salakhutdinov (MLR) pp 6145–54
[73] Kullback S and Leibler R A 1951 On information and sufficiency *Ann. Math. Stat.* **22** 79–86
[74] Rao C R 1992 Information and the accuracy attainable in the estimation of statistical parameters *Breakthroughs in Statistics: Foundations and Basic Theory* (Springer) pp 235–47
[75] Bishop C M and Nasrabadi N M 2006 *Pattern Recognition and Machine Learning* vol 4 (Springer) p 4
[76] Peherstorfer B, Willcox K and Gunzburger M 2018 Survey of multifidelity methods in uncertainty propagation, inference and optimization *SIAM Rev.* **60** 550–91
[77] Giles M B 2015 Multilevel monte carlo methods *Acta Numer.* **24** 259–328
[78] Heinrich S 2001 Multilevel monte carlo methods *Large-Scale Scientific Computing: Third Int. Conf., LSSC 2001 Sozopol* (*Bulgaria, 6–10 June 2001 Revised Papers 3*) (Springer) pp 58–67
[79] 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 *Int. J. Uncertain. Quantification* **14** 21–65
[80] Paszke A *et al* 2019 Pytorch: An imperative style, high-performance deep learning library *Advances in Neural Information Processing Systems* vol 32
[81] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980)
[82] Virtanen P, Gommers R, Oliphant T E, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J and van der Walt S J 2020 SciPy 1.0 Contributors, SciPy 1.0: fundamental algorithms for Scientific Computing in Python *Nat. Methods* **17** 261–72
[83] Nogueira F 2014 Bayesian optimization: open source constrained global optimization tool for python (available at: https://github.com/bayesian-optimization/BayesianOptimization)
[84] Eriksson D, Bindel D and Shoemaker C A 2019 pysot and poap: an event-driven asynchronous framework for surrogate optimization (arXiv:1908.00420)
[85] Picheny V, Wagner T and Ginsbourger D 2013 A benchmark of kriging-based infill criteria for noisy optimization *Struct. Multidisc. Optim.* **48** 607–26
[86] Gray J, Hearn T, Moore K, Hwang J, Martins J and Ning A 2014 Automatic evaluation of multidisciplinary derivatives using a graph-based problem formulation in openmdao, 15th aiaa *ISSMO Multidisciplinary Analysis and Optimization Conf.* (*Atlanta, Georgia, USA*)
[87] Rios L M and Sahinidis N V 2013 Derivative-free optimization: a review of algorithms and comparison of software implementations *J. Glob. Optim.* **56** 1247–93
[88] Stanley A P J and Ning A 2019 Massive simplification of the wind farm layout optimization problem *Wind Energy Sci.* **4** 663–76
[89] Padrón A S, Thomas J, Stanley A P J, Alonso J J and Ning A 2019 Polynomial chaos to efficiently compute the annual energy production in wind farm layout optimization *Wind Energy Sci.* **4** 211–31
[90] Poon Nicholas M K and Martins Joaquim R R A 2007 An adaptive approach to constraint aggregation using adjoint sensitivity analysis *Struct. Multidisc. Optim.* **34** 61–73
[91] Jonkman J, Butterfield S, Musial W, and Scott G 2009 Definition of a 5-mw reference wind turbine for offshore system development National Renewable Energy Lab.(NREL) Golden CO (United States) *Technical Report*
[92] NREL, FLORIS 2019 Version 1.0.0 (available at: https://github.com/NREL/floris)
[93] Jensen N O 1983 *A Note on Wind Generator Interaction* (Risø National Laboratory)
[94] King J, Fleming P, King R, Martínez-Tossas L A, Bay C J, Mudafort R and Simley E 2021 Control-oriented model for secondary effects of wake steering *Wind Energy Sci.* **6** 701–14
[95] Jasa J, Bortolotti P, Zalkind D and Barter G 2021 Effectively using multifidelity optimization for wind turbine design *Wind Energy Sci. Discuss.* **2021** 1–22
[96] Karpatne A, Kannan R and Kumar V 2022 *Knowledge Guided Machine Learning: Accelerating Discovery Using Scientific Knowledge and Data* (CRC Press)