TECHNISCHE UNIVERSITÄT MÜNCHEN

PROFESSUR FÜR KONTINUUMSMECHANIK

Physics-Aware, Bayesian Machine Learning Models for Uncertainty Quantification of High-Dimensional Systems in the Small Data Regime –

Applications in Random Media

Constantin Georg GRIGO

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Prof. Wolfgang POLIFKE, Ph.D.
Prüfer der Dissertation:	1. Prof. Phaedon-Stelios KOUTSOURELAKIS, Ph.D.
	2. Prof. Paris PERDIKARIS, Ph.D.

Die Dissertation wurde am 15.10.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 19.02.2020 angenommen.

Abstract

The computational burden of present-day computer simulations of complex physical systems particularly hampers their usage in *many-query* applications, such as control problems, optimization/design, or forward/inverse uncertainty quantification. Under such computational conditions, a viable approach is to replace the input-to-output mapping of such expensive computer simulations by a cheap to evaluate, but slightly inaccurate computational model called *surrogate* or *metamodel*. In the context of stochastic partial differential equations (SPDEs) describing problems in random heterogeneous media, the inputs to such simulations often correspond to high-dimensional discretizations of short-scale random fields describing, e.g., material properties, boundary conditions, or source terms – a regime where off-the-shelf machine learning models such as Gaussian process (GP) regression or deep learning approaches either fail due to the infamous curse of dimensionality or require a vast amount of training data that are unavailable because of the large computational cost of every forward simulation run.

In such *Tall Data* (high-dimensional, few samples) problems, it is essential to rely on machine learning based surrogate models which are able to extract only the salient features from the high-dimensional, stochastic inputs. At the same time, a maximum amount of a priori information on the underlying physics of the problem should be incorporated into the surrogate model to reduce the space of possible predictions by exclusion of surrogate model outputs that violate basic physical principles.

The present thesis proposes a fully probabilistic machine learning framework for surrogate modeling of flow problems in random heterogeneous media which is in conformity with the above considerations. It consists of a three-component encoder-decoder structure that first finds a low-dimensional, latent, effective representation of the high-dimensional inputs (e.g., material properties) using a small set of feature functions that are automatically selected by sparsity inducing prior models. The low-dimensional representation then serves as the input to a computationally much cheaper coarse-grained model (CGM) based on coarser spatial discretizations and potentially simplified governing equations. Finally, the solution to the CGM is used to reconstruct the original fine-grained model (FGM) response using a fully probabilistic, parametric mapping. As a result, the model is capable of providing accurate, probabilistic predictions for high input dimensions ($d_\lambda \gtrsim 10\,000$) but only few data ($N \lesssim 100$), even under extrapolative conditions, i.e., when tested on data that are very different from the training set. Modern Stochastic Variational Inference (SVI) techniques and state-of-the-art Monte Carlo (MC) methods are used during training and prediction stages. Moreover, a method for automatic adaption of surrogate model complexity is proposed. The computational model complexity is analyzed and the predictive performance is proven by elucidatory numerical examples.

Zusammenfassung

Der hohe Berechnungsaufwand heutiger Computersimulationen von komplexen physikalischen Systemen limitiert deren Einsatz speziell in *many-query* Anwendungen wie Steuerungsproblemen, Optimierung/Design, Fehlerfortpflanzung oder inversen Problemen. Ein brauchbarer Ansatz unter solchen Berechnungsbedingungen ist, die Eingangsgröße-zu-Ausgangsgröße-Abbildung von solch aufwendigen Simulationen durch ein einfach auszuwertendes, jedoch leicht ungenaues Berechnungsmodell, genannt Ersatz- oder Metamodell, zu ersetzen. Im Rahmen von stochastischen partiellen Differenzialgleichungen (SPDEs), die Probleme in zufälligen heterogenen Materialien beschreiben, entsprechen die Eingangsgrößen zu solchen Simulationen oft hochdimensionalen Diskretisierungen von kurzskaligen Zufallsfeldern, die beispielsweise Materialeigenschaften, Randbedingungen oder Quellterme beschreiben – ein Regime, in dem gebrauchsfertige Machine Learning Modelle wie Gauß-Prozesse oder Deep-Learning Methoden durch den berüchtigten "Fluch der Dimensionalität" entweder fehlschlagen oder eine gewaltige Menge an Daten benötigen, die durch die hohen Berechnungskosten jeder Simulationsauswertung nicht verfügbar sind.

In solchen *Tall Data* Problemen (hochdimensional, wenige Datenpunkte) ist es unerlässlich, auf Machine Learning Modelle zu setzen, die dazu in der Lage sind, nur die hervorstechenden Eigenschaften der hochdimensionalen, stochastischen Eingangsgrößen zu extrahieren. Gleichzeitig sollte eine größtmögliche Menge von a priori Information über die zugurundeliegende Physik des Problems in das Ersatzmodell integriert werden, um den Raum möglicher Vorhersagen durch Ausschluss von Ersatzmodell-Ausgangsgrößen, die grundlegende physikalische Prinzipien verletzen, zu verkleinern.

Die vorliegende Dissertation proponiert ein voll probabilistisches Machine Learning Framework zur Ersatzmodellierung von Strömungsproblemen in zufälligen heterogenen Medien, das in Einklang mit den obigen Überlegungen ist. Es besteht aus einer dreikomponentigen Kodierer-Dekodierer-Struktur, die zunächst mittels einer kleinen Menge von Eigenschaftsfunktionen, die automatisch durch sparsity-erzeugende a priori Modelle ausgewählt werden, eine niedrigdimensionale, latente, effektive Darstellung der hochdimensionalen Eingangsgrößen (z.B. Materialeigenschaften) findet. Die niedrigdimensionale Darstellung dient dann als Eingangsgröße für ein numerisch viel einfacheres, grobkörniges Modell (CGM) basierend auf gröberen räumlichen Diskretisierungen und möglicherweise vereinfachten Konstitutivgesetzen. Schließlich wird die Lösung des CGMs benutzt um die Lösung des ursprünglichen, detailgenauen Modells (FGM) mithilfe einer voll probabilistischen, parametrischen Abbildung zu rekostruieren. Dadurch ist das Modell dazu fähig, genaue, probabilistische Vorhersagen für hochdimensionale Eingangsgrößen ($d_\lambda \gtrsim 10\,000$), aber nur wenige Trainingsdaten ($N \lesssim 100$) selbst unter extrapolativen Bedingungen, d.h., wenn die Testdaten sehr verschieden zu den Trainingsdaten sind, zu liefern. Moderne Stochastic Variational Inference (SVI) Techniken und neueste Monte Carlo (MC) Methoden werden während der Trainings- und Vorhersagephasen verwendet. Außerdem wird eine Methode für automatische Anpassung der Ersatzmodell-Komplexität vorgeschlagen. Die Berechnungskomplexität des Modells wird analysiert und die Vorhersageleistung wird durch einleuchtende numerische Beispiele unter Beweis gestellt.

Acknowledgements

First and foremost, I would like to express my deep gratitude to my supervisor Prof. Phaedon-Stelios Koutsourelakis for giving me the privilege to be part of his research group and for the tremendous efforts he expended to help me bringing forth my research. You have been a reliable source of new ideas whenever I was stuck on a problem. I learned a lot from you, not only subject-related things.

I would like to thank Prof. Paris Perdikaris for being part of the committee, but also for chairing so many conference sessions I was allowed to attend. Your research is inspiring, and the way you are presenting it both in talks and publications is very pleasant.

Sincere thanks are given also to Prof. Wolfgang Polifke who is voluntarily chairing the thesis committee.

Furthermore, I would like to thank all my friends and current or former colleagues from the continuum mechanics group: Isabell Franck for being a role model as the first student who graduated at our group. Markus Schöberl for being a comrade in suffering and friend for so long. Max Rixner for bearing with me in our office and sharing his USB-C cable with me – and especially for switching to a noiseless keyboard ;-). Furthermore, I would like to thank Luca Berardocco for his back wheel on all the bicycle tours we made, and for the enjoyable billiard matches we had. I thank Sebastian Kaltenbach and Jonas Nitzler for the pleasant time during lunch and the coffee breaks, even when the canteen food was horrible. Thanks a lot also to our secretaries Cigdem Filker and Sigrid Harnauer who really helped me with all the paper work during all my years in the group.

Many thanks go to all my friends who made the last four years much more pleasurable. Special thanks go to Chris for having me as a flatmate, Stephan for being my official mentor, Marius for the parties, Stefan for the "Lappentraining", and Max for the frequent after work dinners and cocktails.

Finally, I would like to thank my parents for their constant and unconditional support since I came into the world until today. You have made all this possible for me.

Contents

[1]This section is based on [87], Section 2.2.
[2]This section is based on [104], Appendix A.

Chapter 1

Introduction

1.1 Motivation and related work

Thanks to the persistent advances in both cost efficiency and power of high-performance computer hardware over the last decades, it is nowadays very common for companies or research institutions to spend huge amounts of resources into the development of highly elaborate computer simulations of complex physical or engineering systems. Nevertheless, due to superlinear complexity scaling of most computer codes with system size, resolution etc., there still are (and always will be) simulations which need weeks or even more to complete a single run. Doing tasks like sensitivity analysis, (stochastic) optimization/design, uncertainty propagation, or inverse problems, where such simulations need to be carried out a large number of times, is then well-nigh impossible. Researchers and application engineers are therefore eager to find a way to alleviate or circumvent this heavy computational burden.

In the context of random heterogeneous media [1], many problems are defined by stochastic partial differential equations (SPDEs) with, e.g., random coefficients, forcing terms, boundary conditions, or geometry. The randomness in these quantities can often be described by theoretically infinite-dimensional, spatio-temporally correlated random fields. Example problems include heat transfer, wave propagation, mechanical deformation, reaction-diffusion systems – or fluid flow, which is the paradigm of the present work. Closed-form solutions to any of the above problems are hardly ever available and numerical approximations require discretizations that are fine enough to sufficiently resolve all short-scale variabilities, which in turn requires a high amount of computational power for every forward model evaluation. Moreover, accurate discretization of the above mentioned random fields can lead to very high-dimensional random vectors that play the role of uncertain input parameters to the expensive computer simulation defined by a fine scale numerical solver of the PDE. Due to its fine scale nature, the expensive computer model that is to be replaced by the surrogate is referred to as the *fine grained model* (FGM) for the rest of this work. Given the uncertainties in the high-dimensional input parameters, one is often interested in how these uncertainties translate to uncertainties in the FGM output – a typical expression of the forward *uncertainty quantification* (UQ) or *uncertainty propagation* (UP) problem.

UQ is a matter of particular interest not only in the research area of random heterogeneous media. In general, it is the quantitative reasoning that aims to estimate the deviation of predictions of a *model* to the actual ground truth [2–4]. In more mathematical terms, a *model* can be thought of as a function $\hat{\boldsymbol{u}} : \Lambda \mapsto U$, i.e., every input $\boldsymbol{\lambda} \in \Lambda$ is mapped to an output $\boldsymbol{u} = \hat{\boldsymbol{u}}(\boldsymbol{\lambda}), \boldsymbol{u} \in U$. The function $\hat{\boldsymbol{u}}(\boldsymbol{\lambda})$ is not necessarily given in an analytical form, but pointwise evaluation is always possible by, for example, carrying out a physical experiment or running a computer code just as the expensive FGM simulations mentioned above. Uncertainties in $\boldsymbol{u}$ may either arise due to the fact that the model is an inaccurate description of the underlying process (model form uncertainty), or because the input parameters $\boldsymbol{\lambda}$ are uncertain themselves (parametric uncertainty), either due to an inherent random process (aleatoric) or due to lack of knowledge (epistemic). Formally, uncertain model inputs $\boldsymbol{\lambda}$ are associated with a probability space $(\Lambda, \mathcal{S}, p)$ where $\mathcal{S}$ is a set of events over Λ (σ-algebra) and p is a probability measure assigned to the events. Typical UQ problems can roughly be assigned into two different classes: The forward UQ or UP problem [5], or the *inverse (UQ) problem* [6, 7]. Inverse problems can most generally be described as the task of finding the most plausible distribution of input parameters $\boldsymbol{\lambda}$ that leads to a certain set of response samples $\mathcal{D}_{\boldsymbol{u}} = \left\{ \boldsymbol{u}^{(n)} \right\}_{n=1}^{N}$. Such problems typically arise in settings where the quantity of interest can not be measured directly, but only estimated from observations of a different quantity. Fields where inverse problems play an important role include geophysics [8], imaging [9], heat transfer [10], source identification [11] or biomechanical applications [12, 13].

The goodness of solution to any forward and inverse UQ problem is sensitively dependent on the number of data, i.e., the number of model evaluations $\boldsymbol{u} = \hat{\boldsymbol{u}}(\boldsymbol{\lambda})$ that are available. However, any forward model evaluation typically entails a certain amount of hindering cost, such as CPU time or required memory. Under such conditions, it is a popular approach to construct much cheaper, although less accurate surrogate models based on usually only a small number of forward evaluations of the original, expensive computer simulation. The surrogate model may then be used to carry out any UQ or, in general, any other multi-query or performance critical task at the expense of inaccuracies in the final results.

Surrogate models that are popular in the context of UQ and the solution of SPDEs are, e.g., (generalized) Polynomial Chaos expansions (gPC) [14–16] (see Section 4.5.3) which have a quite long-standing history [17] and were successfully applied to, e.g., sensitivity analysis [18, 19], uncertainty propagation [20, 21], or inverse problems [22, 23]. However, despite substantial progress in (sparse grid) stochastic collocation methods [24–27], these methods typically fall short for problems where the stochastic input dimension d_λ is large, since the gPC coefficients are computed using collocation schemes of one-dimensional Gaussian quadrature points. The number of coefficients/collocation points then naturally blows up exponentially due to tensor product rules, and so does the number of required forward model evaluations to find the respective coefficients. The most popular remedy to this issue in the context of gPC is dimension reduction via truncated Karhunen-Loève expansions/PCA of the high-dimensional stochastic inputs [14, 28]. On the other side, the input parameters of problems in random heterogeneous media are often very short-ranged, highly heterogeneous random fields for which PCA variances decay slowly, in which case the inputs remain considerably high-dimensional even after dimension reduction. Apart from their poor scaling to high dimensions, standard gPC formulations are non-Bayesian, i.e., they only yield non-probabilistic point estimates for their expansion coefficients, thereby restraining the associated uncertainties due to limited data.

A further, widely used surrogate modeling technique very popular for the solution of SPDEs is given by the reduced-basis (RB) method [29–32]. The reduced-basis method (see Section 4.5.4) is traditionally separated into *offline* and *online* stages, which is the community wording for data generation + model training (offline stage) and prediction phases (online stage). In the offline phase, a set of FGM output vectors, called *snapshots*, is generated and the principal output directions are found either by the so-called greedy algorithm [33, 34] or a singular value decomposition (SVD) [31, 32] of the snapshot matrix. Solutions are then sought in the space spanned by the L principal output directions. The coefficients of the reduced solution are found either by a Galerkin projection [35–38] or, as in very recent approaches, by solving a regression problem with, e.g., Gaussian Processes [39] or deep learning models [40, 41]. In the RB method, however, Galerkin projections often only offer limited computational efficiency benefits and may suffer from numerical instabilities [42, 43]. On the other side, regression-based approaches using standard machine learning models generally struggle with the high input dimensions given by discretized short-scaled random fields. A further flaw is that the reduced basis and the associated coefficients are learned separately in two distinct computations, but should be found jointly as part of a single model as is done, e.g., in deep image-to-image regression problems [44–46].

A more recent strategy is to view surrogate modeling of complex physical systems/SPDEs directly as supervised machine learning tasks that can be solved by standard tools such as, e.g., Gaussian process (GP) regression [47, 48] (see Section 4.5.2). Roughly speaking, Gaussian processes can be viewed as the generalization of finite-dimensional multivariate normal distributions to the infinite-dimensional space of continuous functions. Just as for Gaussian distributions, conditional densities can be found in closed form and be used to predict unobserved variables from observed data. Due to their analytical tractability and their built-in uncertainty quantification, GP regression models enjoy great popularity among engineers and applied mathematicians and have been widely used as surrogates for problems in random media, e.g., for porous medium flow [49], structural mechanics [50], source identification [11], or other purely academic examples [51]. A further attractive feature of supervised learning with Gaussian processes is the possibility to seamlessly merge multi-fidelity data by learning cross-correlations of the variable fidelity models [52–60]. Despite recent progress in automated dimension reduction [61] and leveraging underlying physics directly in the GP covariance function [62], poor scaling behavior with stochastic input dimension and their $\mathcal{O}(N^3)$ complexity with training data N are major drawbacks of GPs for regression of high-dimensional systems.

Particularly driven by the outstanding progress of deep learning methods [63, 64] in computer vision [65–67] and generative modeling [68, 69] and the free availability of sophisticated software packages [70, 71], artificial neural networks (ANNs, see Section 4.5.1) have found their way into surrogate modeling of complex physical systems simulations [46, 72–76] as well. The most basic version of an ANN, the *multilayer perceptron* or *feed-forward neural network* [77, 78], is a mathematical function of alternating linear combination and nonlinear activation of excitations of nodes that are arranged in layers, constructed with the basic idea to imitate biological neural networks. Over the last decades, many new kinds of different "layers" and network architectures have been developed, such as convolutional layers [65] for shift-invariant inputs (e.g., image data), recurrent layers/networks [79] with an internal state/memory for use with sequential inputs, or pooling and dropout layers [80] for dimension reduction and regularization. The common characteristic of deep learning models is their stacked structure of layers and activations. Their theoretically unlimited model complexity makes them particularly well suited for problems with massive datasets, which is another reason for their increasing popularity. On the other side, surrogate modeling is by definition a *Small Data* problem, a regime where excessive model complexity of deep learning models can be hindering. A very promising approach to alleviate the issue of small data and overly large model complexity in surrogate modeling of PDE systems is to augment the loss/likelihood function by the PDE residual, thereby obtaining what is called a "physics-informed" or "physics constrained" neural network [62, 81–85]. In [73, 86], this approach is taken to the extreme case of no supervised data at all, with a loss function that is only based on the governing equations of the physical problem. Although these approaches are encouraging, there are still some unresolved issues for deep learning methods in surrogate modeling of PDE-based problems. For physics-informed neural networks, much of the computational burden is transferred from data generation to repeated evaluation of the PDE residual. For ANNs in general, model training can become very intricate because the loss function is typically not a convex function of the neural network parameters, and gradients easily run into the risk to become vanishingly small, which may severely slow down model training. Moreover, there is still no automated method to find the optimal network architecture (i.e., number, size, type of layers, etc.) for a given learning problem.

In conclusion, all of the currently most popular surrogate modeling techniques have their own troubles and drawbacks when applied in the *Tall Data* (high dimensions, few data) regime prevailing in SPDE-based problems in random heterogeneous media. Some models are lacking fully probabilistic formulations (RB methods, gPC), are often hard to train (ANNs, GP regression), or may not offer a very large computational benefit (RB methods). But most notably, however, is the fact that most of the above models do not perform sufficiently well given high-dimensional stochastic inputs (gPC, GP regression, vanilla ANNs). The main reason for this is that direct integration of available a priori knowledge about the underlying physical system into these models is difficult or not even possible (except for physics-informed ANNs).

The basic motivation behind this thesis is therefore to develop a machine learning framework for surrogate modeling of numerical PDE solvers for problems in random heterogeneous media that yields accurate predictions even when

- the input dimension is high, i.e., when we are faced with the "curse of dimensionality" – in the problems of this work, we consider input data corresponding to discretized material properties/microstructures of effective dimensionality $d_\lambda \gtrsim 10\,000$; and
- the number of training data N is small, i.e., the number of fine grained model (FGM) input/output pairs that can virtually be afforded to train the machine learning surrogate is not more than $N \lesssim 100$ samples.

To achieve optimal model performance even under this *Tall Data* setting, it is vital (a) to extract only a small number of microstructural features from the high-dimensional, random inputs that contain a maximum amount of information about the FGM response; and (b) to set up a surrogate model architecture that retains as much structure as possible from the underlying FGM problem [87]. While point (a) aims for effective reduction of the high stochastic input dimension, point (b) should exclude surrogate model outputs that clash with fundamental laws of physics, thereby reducing the space of possible surrogate model predictions to physically plausible solutions only.

One way to incorporate information about the underlying physics of the FGM is to make use of lower fidelity, reduced-order models (ROMs) that approximate the original, fine-grained system and are based on, e.g., coarser spatio-temporal discretizations. The standard approach to treat such multi-fidelity sources of information is to generate additional data of input/output pairs of the cheap-to-evaluate, lower fidelity ROM(s) and to merge the information content of both high- and low-fidelity data in a suitable machine learning model, as is done in, e.g., the multi-fidelity Gaussian process[52–60]. Storing and manipulating large datasets of high-dimensional, low-fidelity data may, however, become time and memory consuming as well – let alone the $\mathcal{O}(N^3)$ complexity scaling of Gaussian processes for cases where the number of low-fidelity samples is large.

Rather than treating the information from low-fidelity models as additional training data, another approach is to encase low-fidelity solvers directly into the machine learning based surrogate model, learning a direct mapping between low- and high-fidelity outputs (and potentially inputs as well). This approach is advocated in, e.g., [88, 89]. In this setting, it is of crucial importance that the low-fidelity model(s) are cheap to evaluate, because they need to be called in the online stage every time a new prediction is queried. To estimate the surrogate model uncertainty, the low-fidelity model needs to be evaluated even multiple times to estimate the output statistics given only a single original, high-fidelity input.

In the solution of PDEs with finite elements (FE, see Section 3.1), it is a well known fact that solvers operating on coarser spatio-temporal discretizations provide approximate solutions at a lower computational cost – simply because the system of algebraic equations to be solved is smaller. Since the numerical complexity of finding the solution to a generic linear system of equations may scale with the number of equations as $\mathcal{O}(N_{\text{eqns}}^3)$, the computational effort to solve a much coarser discretized FE model is strikingly smaller. Moreover, it is not even necessary that the low-fidelity model shows only small deviations from the high-fidelity solver – the only necessity is a statistical dependence that can be learned by the machine learning model.

Complementary to the computational simplification achieved by coarser spatio-temporal discretizations, physical processes often exhibit less complex constitutive behavior when investigated on larger length and time scales. This can be exploited to construct coarse-grained low-fidelity solvers of even higher computational efficiency. On the other side, one needs to bear in mind that multi-fidelity codes based on multi-scale considerations often require dissimilar descriptions of the respective input parameters. In our use case of flow problems in random heterogeneous media, it is by no means clear how to describe the input material property random field (permeability) on the coarse scale given a certain fine scale microstructure consisting of a binary material with permeable fluid and impermeable solid spaces. In general, no closed-form solutions to this coarse-graining problem exist and approximation procedures are developed in the field of homogenization theory [1, 90–92]. Moreover, random media are typically described only up to a stochastic level. Also, the (non-analytical) microstructure-to-effective property mapping is non-deterministic by nature, requiring a fully probabilistic treatment of the problem. A data-driven, numerical approach to the homogenization problem appears viable and can be seen as a by-product of the surrogate modeling framework suggested in this thesis.

1.2 Contribution

In consequence of the above considerations, the present work advocates a surrogate modeling framework for problems in random heterogeneous media which at its core unit uses a low-fidelity coarse-grained model (CGM) based on coarser spatial discretizations and potentially simplified governing equations. The CGM serves as a stencil that retains the physical characteristics of the expensive fine-grained model (FGM) and automatically excludes surrogate model predictions that are not in accordance with the coarse-scale behavior of the FGM problem. In the numerical examples presented in this work, the FGM itself corresponds to a fine-scale finite element simulation of fluid flow in random heterogeneous media. The parametric input to the FGM is a high-dimensional, fine scale description of the microstructure of the random medium. The output is given by the fine scale pressure field response.

The inputs to the CGM correspond to a coarse-grained, effective material property field of much lower dimension than the initial, fine-scale description of the material microstructure. The coarse-graining procedure is carried out in a fully probabilistic way and consists of a Gaussian linear model with a large library of basis/feature functions defined in or inspired by the rich literature on random heterogeneous media [1], enriched with quantities from image analysis [93], fluid dynamics [94–97] and other physics [98], or autoencoder representations [99–101]. Sparsity prior models are applied to pick out only a small number of feature functions that appear most predictive for reconstruction of the FGM output. As the effective material property field is treated as a low-dimensional latent variable, these sparsity priors need to be specifically adapted for application in such latent variable models.

The CGM equation system is assembled and solved efficiently by affine decompositions [31] and the application of a sparse banded solver [102]. Finally, the original FGM response is predicted by a Gaussian linear projection that takes into account the spatial nature of the problem defined by the fact that both CGM and FGM responses correspond to spatial fields. Predictions are fully probabilistic, that means that the surrogate model is capable to quantify the uncertainty originating from both limited training data as well as finite model complexity, i.e., information loss during the coarse-graining process.

A fully Bayesian approach is pursued (Section 5.3.4) such that no hyperparameters are left to be tuned by the user. Efficient training and prediction algorithms are developed that make use of modern Monte Carlo or Variational Inference techniques to compute latent variable expected values. These are based on first order derivatives of the CGM that are found by solving the adjoint problem. Both coarse-graining/dimension reduction and surrogate modeling/reconstruction steps are trained in a joint fashion, such that the encoded, low-dimensional, latent representation of the stochastic fine-grained inputs contains a maximum amount of information for reconstruction of the FGM response, and not for reconstruction of the input itself, as is the case for classical dimension reduction methods.

The paradigmatic applications used in this work are flow problems in random heterogeneous media. One set of examples investigates Darcy flow FGM data in binary random permeability fields drawn from a level-cut Gaussian process. Another set of examples examines an FGM given by a Stokes flow FEM simulation in random porous media, where the microstructure is described by a unit square domain perforated by polydisperse spherical exclusions. In all examples, the CGM is based on a Darcy flow FEM simulation discretized on much coarser grids than the fine-grained models, motivated by the limiting case of Stokes/Darcy equivalence for vanishing characteristic length scales of the fine scale microstructure.

With the presented surrogate modeling approach, a speedup up to a factor of $\approx$ 15 000 is achieved with very high predictive accuracy (coefficient of determination $R^2 \gtrsim 0.9$) for considerably small FGM datasets of only $N \lesssim 100$. The homogenization problem, i.e., finding effective material properties to a given microstructure, is solved in a numerical way and (approximate) predictive posteriors are computed. Moreover, it is possible to imprint information from boundary conditions or forcing terms on the CGM. This makes the model much more apt than standard regression approaches (e.g., ANNs or GPs) under *extrapolative* conditions, i.e., when the test data are very different to the training data due to variation of boundary conditions/forcing terms. A refinement procedure is suggested to increase model complexity in an optimal way.

The findings of this work have been published in

C. Grigo, P.-S. Koutsourelakis:

"A physics-aware, probabilistic machine learning framework for coarse-graining high-dimensional systems in the Small Data regime",
Elsevier Journal of Computational Physics 2019, Volume 397, 108842

C. Grigo, P.-S. Koutsourelakis:

"Bayesian Model and Dimension Reduction for Uncertainty Propagation: Applications in Random Media",
SIAM/ASA Journal on Uncertainty Quantification 2019 7:1, 292-323

C. Grigo, P.-S. Koutsourelakis:

"A data-driven model order reduction approach for Stokes flow through random porous media",
Proc. Appl. Math. Mech. 2018, 18: e201800314

C. Grigo, P.-S. Koutsourelakis:

"Probabilistic reduced-order modeling for stochastic partial differential equations",
Eccomas Proceedia UNCECOMP (2017) 111-129

1.3 Outline

This thesis starts with a recapitulation of all the theory and methods that are necessary for the understanding of the proposed physics-aware surrogate modeling framework and its application to the paradigmatic problem of flow through random heterogeneous media. The governing equations that are most commonly used to describe fluid flow through random media, namely Stokes and Darcy flow, are introduced in Chapter 2 together with a description of how realistic random binary microstructures can be generated/reconstructed by computer simulation.

Subsequently, Chapter 3 gives a short introduction to the notion of *stochastic partial differential equations* (SPDEs) and the primary technique to the numerical solution of partial differential equations, namely the *finite element method* (FEM).

Next, Chapter 4 contains a zero-base introduction to probability theory, machine learning, and uncertainty quantification starting from linear regression and Bayes' law. The chapter gives a thorough description of the all methods that are used in the proposed physics-aware machine learning framework such as sparsity priors, efficient Monte Carlo techniques, Variational Inference, and gradient-based stochastic optimization. Moreover, a summary of the most important surrogate modeling approaches to the solution of SPDEs that are in direct competition with the proposed method is given together with short analyses of the respective advantages and disadvantages.

Chapter 5 represents the core element of this thesis and contains an in-depth description of the suggested physics-aware machine learning framework for high-dimensional systems. An outline of the model architecture is given, the prior models that have been used are derived and efficient training algorithms are presented. Moreover, an algorithm to generate predictive samples is showcased, a numerical complexity analysis for both training and prediction stages is performed and a method to automatic adjustment of model complexity is suggested.

The numerical experiments published in [87, 103, 104] are presented in Chapter 6. These contain examples where the coarse-grained model (CGM) is based on both identical (Section 6.1) and simplified (Section 6.2) governing equations compared to the fine grained model (FGM) data. In particular, validation examples are carried out to reproduce closed-form solutions, the predictive performance is evaluated in dependence of the number of training data and the coarse-grained model complexity, effective material parameters are showcased and extrapolative capabilities are given proof by using data based on different boundary conditions. Moreover, a simple uncertainty propagation problem indicates the trivial coupling of the proposed surrogate to any kind of multi-query applications.

Chapter 7 concludes this work with a review of the main achievements of this thesis and presents a list of extensions and research directions that are worth to be investigated in the future.

Chapter 2

Flow through random heterogeneous media

As noted in the previous chapter, the paradigmatic application for the proposed surrogate modeling framework is given by flow problems in random heterogeneous media. Ultimately, the method can be contemplated for many other problems as well, such as, e.g., solving Maxwell's equations in random permittivity fields [105], the Helmholtz equation with stochastic source fields [106, 107], or deformation problems with random material elasticity [108]. However, flow problems in random media, particularly those based on the Poisson equation describing Darcy flow, are among the most popular choices for trying and testing new surrogate modeling or uncertainty quantification methods (see, e.g., [20, 31, 46, 55, 72, 109]) and are therefore investigated also in this work. For comprehension of the proposed surrogate modeling framework, the following two chapters are not absolutely mandatory to understand.

Flow through random heterogeneous media describes the behavior of fluids when passing through a body with highly varying material properties on short length scales ℓ_f, as for example in a sponge, a porous rock, or a sandy soil. As depicted in Figure 2.1, the substructure length scale ℓ_f is much smaller than the size $L = 1$ of the domain $\Omega = [0, 1]^2$ which is representative for the size of a typical engineering component. In random porous media, flow velocities $\boldsymbol{V}$ are slow so that viscous forces are predominant compared to inertial forces, leading to low Reynolds numbers

$$\mathrm{Re} = \frac{\rho|\bar{\boldsymbol{V}}|\ell_f}{\mu} \ll 1, \tag{2.1}$$

defining the so-called *creeping flow* regime, where ρ denotes the fluid density, $|\bar{\boldsymbol{V}}|$ the volume averaged absolute fluid velocity and μ is the dynamic fluid viscosity.

Depending on the length scales, the exact material properties, the desired accuracy and the available computational resources, flow through random media is modeled using different constitutive assumptions which will be discussed in the subsequent sections.

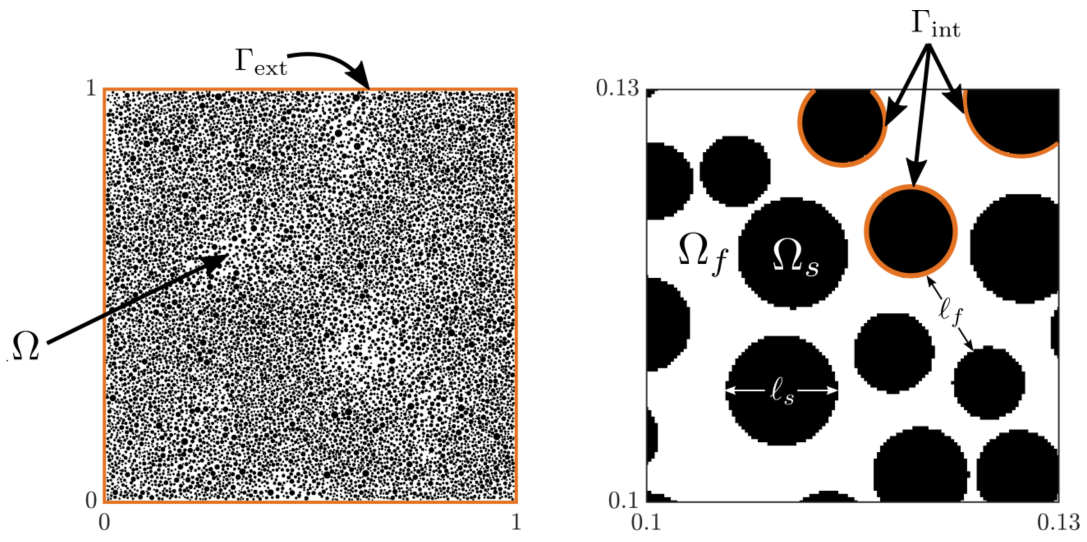


FIGURE 2.1: Sample of a porous medium microstructure with non-overlapping, randomly distributed spherical exclusions (black circles). Fluid flow happens in the pore space Ω_f (white area). The left side depicts a whole microstructure on a unit square domain, whereas the right side is zoomed in for clarity of notation. Picture taken from [87].

The most general form for the equations of motion of an incompressible Newtonian fluid are the famous *Navier-Stokes equations*

$$\rho\left(\frac{\partial}{\partial t}\boldsymbol{V}+\boldsymbol{V}\cdot\nabla_x\boldsymbol{V}\right)=-\nabla_x P+\mu\nabla_x^2\boldsymbol{V}+\rho\boldsymbol{f} \qquad \text{for} \quad \boldsymbol{x}\in\Omega_f, \tag{2.2a}$$

$$\nabla_x\cdot\boldsymbol{V}=0 \qquad \text{for} \quad \boldsymbol{x}\in\Omega_f, \tag{2.2b}$$

where Ω_f is the fluid domain (see Figure 2.1), P is the fluid pressure field and $\boldsymbol{f}$ is an external weight force. Except for a few isolated cases, the nonlinear Navier-Stokes equations are generally not solvable in closed form and numerical simulations can become extremely cumbersome, especially for turbulent systems. It is thus imperative to find suitable simplifications that enable the application of efficient numerical solvers whilst accurately describing the physics of the simulated experiment. In the creeping flow regime, such simplifications are given by *Stokes* and *Darcy flow* which are introduced in the following.

2.1 Stokes flow

The equations of motion pertaining to Stokes flow, the so-called Stokes equations, can be derived from the Navier-Stokes equations for an incompressible Newtonian fluid given by Equation (2.2) assuming small Reynolds numbers $\mathrm{Re} \ll 1$. To do so, observe that every term of Equation (2.2a) has the units of a body force, i.e., $\frac{\text{force}}{\text{volume}}$ or $\frac{\text{mass}}{\text{length}^2\cdot\text{time}^2}$. To obtain an equation that is independent of the physical scale of the problem but only depends on the Reynolds number Re, Equation (2.2a) is multiplied by $\frac{\ell_f}{\rho \bar{V}^2}$ and we introduce the dimensionless quantities

$$V' = \frac{V}{\bar{V}}, \quad P' = \frac{P\ell_f}{\mu \bar{V}}, \quad f' = \frac{\rho \ell_f^2}{\mu \bar{V}} f, \quad \frac{\partial}{\partial t'} = \frac{\ell_f}{\bar{V}} \frac{\partial}{\partial t'}, \quad \nabla'_x = \ell_f \nabla_x \tag{2.3}$$

to get

$$\frac{\partial}{\partial t'} V' + V' \cdot \nabla'_x V' = -\frac{1}{\mathrm{Re}} \nabla'_x P' + \frac{1}{\mathrm{Re}} \nabla'^2_x V' + \frac{1}{\mathrm{Re}} f' \qquad \text{for} \quad x \in \Omega_f, \tag{2.4a}$$

$$\nabla_x \cdot V' = 0 \qquad \text{for} \quad x \in \Omega_f. \tag{2.4b}$$

From Equation (2.4a) it is obvious that for Re $\ll 1$, the left hand side can be neglected and, dropping the primes and multiplying by the Reynolds number Re, we obtain the *Stokes equations* of momentum and mass conservation,

$$\nabla_x P - \nabla_x^2 V = f \qquad \text{for} \quad x \in \Omega_f, \tag{2.5a}$$

$$\nabla_x \cdot V = 0 \qquad \text{for} \quad x \in \Omega_f, \tag{2.5b}$$

which form an appropriate constitutive law for the creeping flow regime Re $\ll 1$. Together with a proper set of boundary conditions

$$V = V_{bc} \qquad \text{for} \quad x \in \Gamma_V, \tag{2.5c}$$

$$t = (\nabla_x V_{bc} - P_{bc} I) n \qquad \text{for} \quad x \in \Gamma_P, \tag{2.5d}$$

where t is a Cauchy traction vector and n the unit outward normal, Equation (2.5) forms a linear boundary value problem which can be solved by a standard finite element approach discussed in Section 3.1.2.

Stokes flow through random porous media is modeled assuming a porous domain $\Omega = \Omega_f \cup \Omega_s$ composed of the two disjoint sets $\Omega_f \cap \Omega_s = \emptyset$, where Ω_f denotes the pore or fluid space and Ω_s defines the impermeable solid phase of the material. The equations of motion given by Equation (2.5) are only defined on Ω_f which needs to be topologically fully connected to yield a unique solution.

In 2D, a fully connected pore space Ω_f can be realized, for instance, by non-overlapping polydisperse spherical exclusions (Section 2.4.2), as indicated by the black dots in Figure 2.1. On the internal solid-fluid interface Γ_{int}, boundary conditions need to be specified. The most common choice is the no-slip condition

$$V = \mathbf{0} \qquad \text{for} \quad x \in \Gamma_{\text{int}}, \tag{2.5e}$$

whereas the external boundary $\Gamma_{\text{ext}} = \partial\Omega = \partial\Omega_f \backslash \Gamma_{\text{int}}$ is typically split into a part Γ_V with essential boundary conditions where the fluid velocity is prescribed (see Equation (2.5c)) and a natural part $\Gamma_P = \Gamma_{\text{ext}} \backslash \Gamma_V$, $\Gamma_V \cap \Gamma_P = \emptyset$ where surface tractions t are predefined (Equation (2.5d)). Samples for different random porous media and

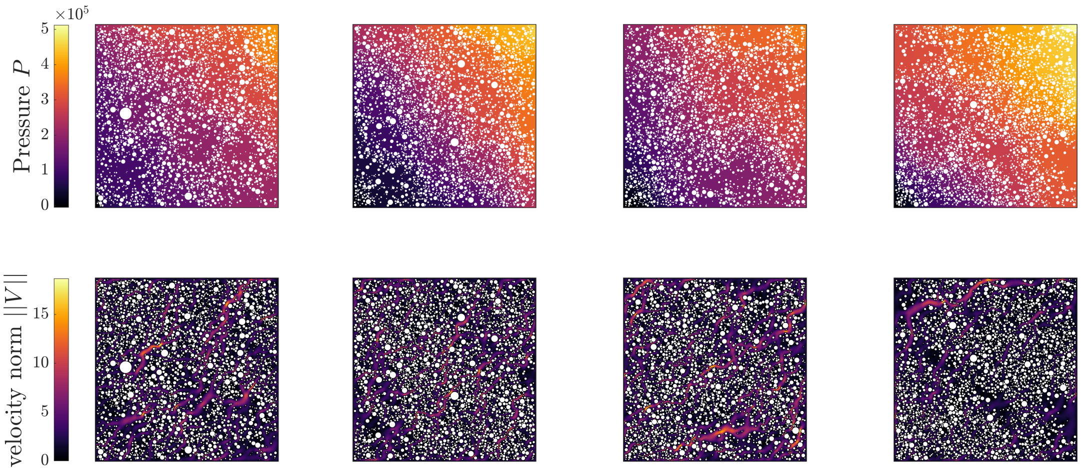


FIGURE 2.2: Pressure field $P(\boldsymbol{x})$ and velocity norm $\|\boldsymbol{V}(\boldsymbol{x})\|$ for Stokes flow through different random porous media with boundary conditions $\boldsymbol{t}(\boldsymbol{x}) = \boldsymbol{0}$ at $\boldsymbol{x} = \boldsymbol{0}$ and $\boldsymbol{V}_{bc}(\boldsymbol{x}) = (1 \quad 1)^T$ at $\partial\Omega \backslash \{\boldsymbol{0}\}$.

the boundary conditions $\boldsymbol{t}(\boldsymbol{x}) = \boldsymbol{0}$ at $\Gamma_P = \boldsymbol{0}$ and $\boldsymbol{V}_{bc}(\boldsymbol{x}) = \binom{1}{1}$ at $\Gamma_V = \partial\Omega \backslash \Gamma_P$ are depicted in Figure 2.2.

2.2 Darcy flow

Darcy's law for fluid flow through porous media was first formulated by Henry Darcy as an empirical law found by experiments that were carried out by Darcy in the context of the creation of an irrigation system for the city of Dijon in 1856 [110]. It is virtually identical to Fourier's law [111] of thermal conduction, Ohm's law of electrical resistance [112] and Fick's law of particle diffusion [113].

Darcy's law states that the flux density, i.e., the volumetric fluid flow per unit area and time $\bar{\boldsymbol{V}}$ (which has the units of a velocity $\frac{\text{length}}{\text{time}}$) is proportional to the negative pressure gradient[1]

$$\bar{\boldsymbol{V}} = -\frac{\boldsymbol{K}}{\mu} \nabla_{\boldsymbol{x}} \bar{P} \qquad \text{for} \quad \boldsymbol{x} \in \Omega \tag{2.6a}$$

where $\boldsymbol{K}$ is a second order tensor called *permeability*. To ensure that fluid is always flowing from high to low pressure regions, it is indispensable that $\boldsymbol{K}$ is positive definite and that Equation (2.6a) includes a minus sign. Additionally to Darcy's law, the incompressibility condition holds

$$\nabla_{\boldsymbol{x}} \cdot \bar{\boldsymbol{V}} = 0 \qquad \text{for} \quad \boldsymbol{x} \in \Omega \tag{2.6b}$$

[1]The bar over both quantities will become clear in the following subsection.

and together with boundary conditions of the form

$$\bar{\boldsymbol{V}} \cdot \boldsymbol{n} = \boldsymbol{V}_{bc} \cdot \boldsymbol{n} \qquad \text{for} \quad \boldsymbol{x} \in \Gamma_V, \tag{2.6c}$$

$$\bar{P} = P_{bc} \qquad \text{for} \quad \boldsymbol{x} \in \Gamma_P \tag{2.6d}$$

defines an elliptic boundary value problem which is probably the most studied system in the context of finite elements and stochastic partial differential equations (SPDEs).

Darcy flow through random porous media is modeled assuming a location dependent permeability tensor $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{x})$ that can either be continuously varying over the domain Ω (if some local homogenization of the material has been performed before) or it can be binary of the form

$$\boldsymbol{K}(\boldsymbol{x}) = \boldsymbol{K}_s \mathbb{1}_s(\boldsymbol{x}) + \boldsymbol{K}_f \mathbb{1}_f(\boldsymbol{x}) \tag{2.7}$$

where $\mathbb{1}_s, \mathbb{1}_f$ are the indicator functions for the solid and fluid phases, respectively, and $\boldsymbol{K}_s, \boldsymbol{K}_f$ are the corresponding permeability tensors.

Modeling Darcy flow assuming a binary diffusivity field as defined above may be inaccurate because, as was shown theoretically in [114], Darcy flow arises from Stokes flow through porous media by homogenization over a suitably sized representative volume element (RVE). If the size r_0 of the RVE, i.e., the homogenization radius, is notably smaller than the size $L = 1$ of the domain Ω/the engineering component, then flow through porous media is accurately modeled by Darcy flow assuming a *continuous* permeability field $\boldsymbol{K}(\boldsymbol{x})$ that is varying on a length scale r_0.

Nevertheless, finding a viable surrogate model as presented in Section 5 for the elliptic PDE defined by Equation (2.6) with highly varying, binary or continuous permeability field $\boldsymbol{K}(\boldsymbol{x})$ may still be helpful: As mentioned at the beginning of this section, exactly the same PDE forms the constitutive law for a variety of physical phenomena, and for some of them (e.g., for thermal conduction) the description of a highly varying, binary coefficient field $\boldsymbol{K}(\boldsymbol{x})$ is much closer to the true physical picture than for flow through porous media.

2.3 The Stokes to Darcy limit[2]

Darcy's law for flow through random porous media was found empirically in [110] and it took more than a century to find a theoretical derivation from Navier-Stokes equations by homogenization, first for stationary flow in periodic microstructures [115, 116], connected $3D$ solid phase matrix [117], non-stationary flow [118] and non-periodic random media [114].

[2]This section is based on [87], Section 2.2.

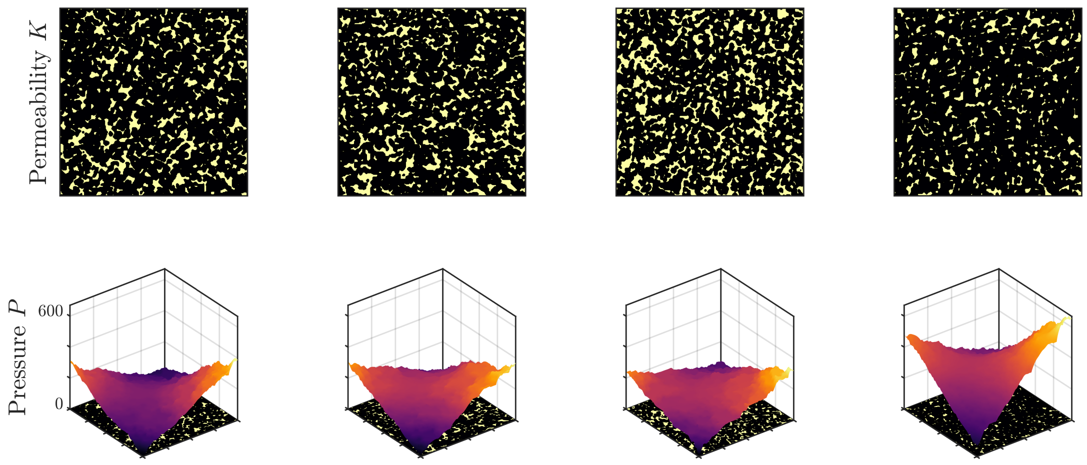


FIGURE 2.3: Binary, isotropic permeability field $\boldsymbol{K}(\boldsymbol{x}) = (1 \cdot \mathbb{1}_s(\boldsymbol{x}) + 500\mathbb{1}_f(\boldsymbol{x}))\boldsymbol{I}$ (top row) and pressure field $P(\boldsymbol{x})$ with boundary conditions $P_{bc} = 0$ at $\Gamma_P = \{\boldsymbol{0}\}$ and $\boldsymbol{V}_{bc} = (800 - 2000x \quad 1200 - 2000y)$ at $\Gamma_V = \partial\Omega \backslash \{\boldsymbol{0}\}$.

In homogenization theory, a boundary value problem is found on an RVE [119–121] of linear size r_0 sufficiently larger than the microstrucutral length scale ℓ_f. Moreover, the volume-averaged quantities, i.e., intrinsic phase averages

$$\bar{P} = \frac{1}{|\Omega_{f,\mathrm{RVE}}|} \int_{\Omega_{f,\mathrm{RVE}}} P(\boldsymbol{x}) dV, \qquad \bar{\boldsymbol{V}} = \frac{1}{|\Omega_{f,\mathrm{RVE}}|} \int_{\Omega_{f,\mathrm{RVE}}} \boldsymbol{V}(\boldsymbol{x}) dV \tag{2.8}$$

with $|\Omega_{f,\mathrm{RVE}}|$ the volume of the fluid part of the RVE, should exhibit sufficient variation over the domain Ω, which means that the linear domain size L needs to be large compared to the RVE size r_0. To sum up, Stokes flow through random porous media can be described by the Darcy constitutive equations Equation (2.6) in the limit of

$$\ell_f \ll r_0 \ll L \tag{2.9}$$

which can be relaxed, according to [114], to the weaker conditions

$$\left(\frac{r_0}{L}\right)^2 \ll 1, \qquad r_0 \gtrsim 5\ell_f \tag{2.10}$$

for correct assumption of Darcy's equations of motion.

Although the Stokes/Darcy equivalence is theoretically proven in the case of infinite scale separation as defined by Equation (2.9), determining an accurate, effective permeability field $\boldsymbol{K}(\boldsymbol{x})$ that goes beyond the scope of rough approximation formulas [95, 96] continues to be a substantial computational task usually approached by solving an RVE subscale problem [119].

It is noted that, even when the scale separation limit Equation (2.9) does not hold, the Darcy equations of motion may still be used as a stencil of a machine learning model as an accurate surrogate for fine-scale Stokes flow simulations. As shown in the experiments of Section 6.2, a potential mismatch in background physics of such a stencil to the data can be compensated to a certain extent by parametric model adaptivity. Moreover, a fully probabilistic surrogate, as developed in this work, will directly indicate an inexpedient model by excessively broad/uninformative predictive distributions.

2.4 Reconstruction of random microstructures

The reconstruction of microstructural samples from limited material information such as lower-order correlation functions is a field of ongoing research (see, e.g., [122–124]) and is therefore only discussed up to the parts necessary for the numerical experiments presented in Section 6. A central technique is the generation of 2D Gaussian random field realizations, i.e., random fields with prescribed second order correlation function.

2.4.1 Generation of 2D Gaussian random field realizations[4]

For the generation of approximate 2D Gaussian process samples [125] (see also Section 4.5.2 for an introduction to Gaussian process regression) with stationary covariance functions, i.e., covariance functions of the form $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x} - \boldsymbol{x}') = k(\boldsymbol{r})$, we use

Theorem 1 (S. Bochner, 1959 [126], quoted from [127]) *"A complex-valued function k on $\mathbb{R}^d$ is the covariance function of a weakly stationary mean square continuous complex-valued random process on $\mathbb{R}^d$ if and only if it can be represented as*

$$k(\boldsymbol{r}) = \int_{\mathbb{R}^d} e^{-i\boldsymbol{w}^T\boldsymbol{r}} s(\boldsymbol{w}) d\boldsymbol{w}, \tag{2.11}$$

where $s(\boldsymbol{w})d\boldsymbol{w}$ is a positive finite measure."

which has been proven in [128]. This means that the covariance function of every stationary Gaussian process can be written as the spectral decomposition given by Equation (2.11), where

$$p(\boldsymbol{w}) = \frac{1}{\alpha} s(\boldsymbol{w}) = \frac{1}{\alpha} \frac{1}{(2\pi)^d} \int k(\boldsymbol{r}) e^{i\boldsymbol{w}^T\boldsymbol{r}} d\boldsymbol{r} \tag{2.12}$$

is a valid probability density with $\alpha = \int s(\boldsymbol{w}) d\boldsymbol{w}$. This can be written as

$$k(\boldsymbol{x} - \boldsymbol{x}') = \alpha \left\langle e^{-i\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{x}')} \right\rangle_{p(\boldsymbol{w})} = \alpha \left\langle \cos(\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{x}')) \right\rangle_{p(\boldsymbol{w})} \tag{2.13}$$

[4]This section is based on [104], Appendix A.

Kernel function $k(r) \propto$	Spectral density $p(w)$	Comments
$\frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{l}\right)^{\nu} I_{\nu}\left(\frac{\sqrt{2\nu}r}{l}\right)$	$\frac{2\sqrt{\pi}\Gamma(\nu+\frac{1}{2})(2\nu)^{\nu}}{\Gamma(\nu)l^{2\nu}} \left(\frac{2\nu}{l^2}4\pi^2 w^2\right)^{-(\nu+\frac{1}{2})}$	Matérn kernel
$\frac{\sin(2\pi r/l)}{r/l}$	$\mathcal{U}(w\|-2\pi/l, 2\pi/l)$	
$\left(\frac{\sin(2\pi r/l)}{r/l}\right)^2$	$\text{tri}(\frac{wl}{2\pi})$	$\text{tri}(x) = \begin{cases} x+1 & x<0, \\ -x+1 & x => 0. \end{cases}$
$\cos(2\pi r/l)$	$\frac{1}{2}\left(\delta(w-l^{-1}) + \delta(w+l^{-1})\right)$	
$\frac{J_1\left(2\pi\sqrt{r_1^2+r_2^2}/l\right)}{\sqrt{r_1^2+r_2^2}/l}$	$\mathcal{U}\left(\sqrt{w_1^2+w_2^2}\|0, 2\pi/l\right)$	$2D$; J_1: 1st Bessel func. of 1st kind

TABLE 2.1: Spectral densities for some $2D$ covariance functions $k(\boldsymbol{r})$. The Matérn kernel contains the Ornstein-Uhlenbeck process ($k(r) \propto e^{-|r|/l}$) for $\nu = 1/2$ and the squared exponential kernel ($k(r) \propto e^{-r^2/l^2}$) for $\nu \to \infty$.

where it was used that both $k(\boldsymbol{x}-\boldsymbol{x}')$ and $p(\boldsymbol{w})$ are real functions. We can write that

$$\begin{aligned} k(\boldsymbol{x}-\boldsymbol{x}') &= \alpha \left\langle \cos(\boldsymbol{w}^T\boldsymbol{x} + b - \boldsymbol{w}^T\boldsymbol{x}' - b) \right\rangle_{p(\boldsymbol{w},b)} \\ &= \alpha \left\langle \cos(\boldsymbol{w}^T\boldsymbol{x}+b)\cos(\boldsymbol{w}^T\boldsymbol{x}'+b) + \sin(\boldsymbol{w}^T\boldsymbol{x}+b)\sin(\boldsymbol{w}^T\boldsymbol{x}'+b) \right\rangle_{p(\boldsymbol{w},b)} \end{aligned} \tag{2.14}$$

which, assuming that $p(\boldsymbol{w},b) = p(\boldsymbol{w})p(b)$ and $p(b) = \mathcal{U}(b|0,2\pi)$, simplifies to

$$\begin{aligned} k(\boldsymbol{x}-\boldsymbol{x}') &= 2\alpha \left\langle \cos(\boldsymbol{w}^T\boldsymbol{x}+b)\cos(\boldsymbol{w}^T\boldsymbol{x}'+b) \right\rangle_{p(\boldsymbol{w},b)} \\ &= 2\alpha \left\langle \sin(\boldsymbol{w}^T\boldsymbol{x}+b)\sin(\boldsymbol{w}^T\boldsymbol{x}'+b) \right\rangle_{p(\boldsymbol{w},b)}. \end{aligned} \tag{2.15}$$

Writing the expected value as a Monte Carlo estimate[5]

$$k(\boldsymbol{x}-\boldsymbol{x}') \approx \frac{2\alpha}{M}\sum_{m=1}^{M} \cos(\boldsymbol{w}_m^T\boldsymbol{x}+b_m)\cos(\boldsymbol{w}_m^T\boldsymbol{x}'+b_m) \qquad \boldsymbol{w}_m, b_m \sim p(\boldsymbol{w},b), \tag{2.16}$$

it can be observed that this is exactly the same covariance function as for the Gaussian linear model

$$g(\boldsymbol{x}) = \sqrt{2\alpha/M}\sum_{m=1}^{M} \gamma_m \cos(\boldsymbol{w}_m^T\boldsymbol{x}+b_m), \qquad \gamma_m \sim \mathcal{N}(0,1), \tag{2.17}$$

i.e., approximate realizations of a Gaussian process with stationary kernel $k(\boldsymbol{x}-\boldsymbol{x}')$ are given by Equation (2.17), where $\gamma_m \sim \mathcal{N}(0,1)$, $b_m \sim \mathcal{U}(0,2\pi)$, $\boldsymbol{w}_m \sim p(\boldsymbol{w})$. To find $p(\boldsymbol{w}) = \frac{1}{\alpha}s(\boldsymbol{w})$, one needs to know the spectral density

$$s(\boldsymbol{w}) = \frac{1}{(2\pi)^d}\int k(\boldsymbol{r})e^{i\boldsymbol{w}^T\boldsymbol{r}}d\boldsymbol{r}. \tag{2.18}$$

The spectral densities for some popular stationary covariance kernels are summarized in Table 2.1. Note that for $k(\boldsymbol{r}) = k_{r_1}(r_1)k_{r_2}(r_2)$, it holds that $p(\boldsymbol{w}) = p_{w_1}(w_1)\cdot$

[5]The cosines can just as well be replaced by sines, see Equation (2.15).

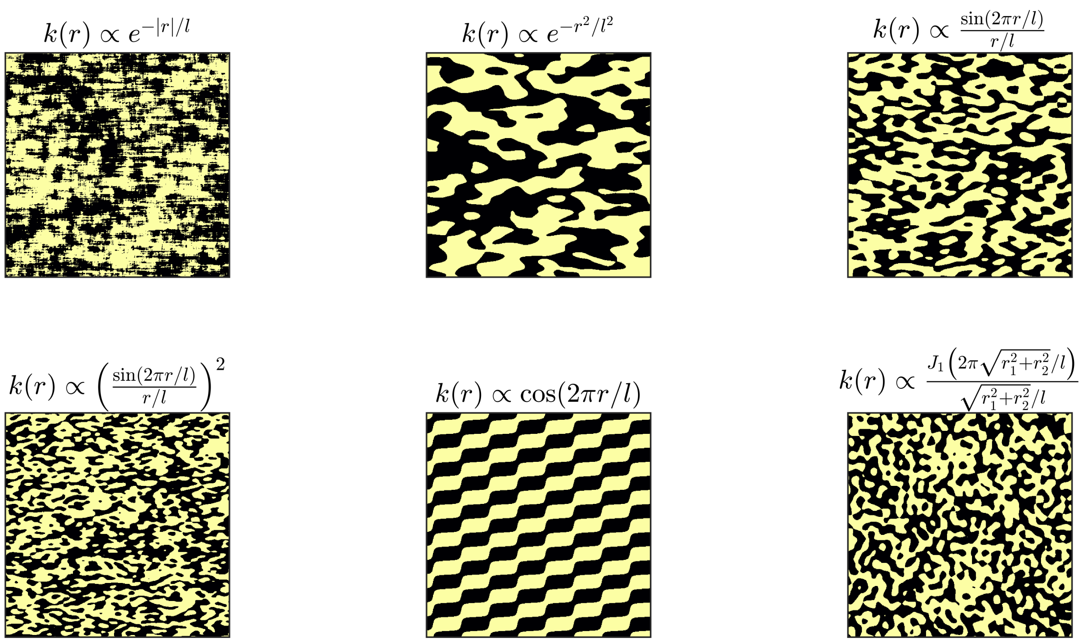

$k(r) \propto e^{-|r|/l}$
$k(r) \propto e^{-r^2/l^2}$
$k(r) \propto \frac{\sin(2\pi r/l)}{r/l}$
$k(r) \propto \left(\frac{\sin(2\pi r/l)}{r/l}\right)^2$
$k(r) \propto \cos(2\pi r/l)$
$k(r) \propto \frac{J_1\left(2\pi\sqrt{r_1^2+r_2^2}/l\right)}{\sqrt{r_1^2+r_2^2}/l}$

FIGURE 2.4: Microstructural samples with expected volume fractions $|\Omega_f| = |\Omega_s| = 0.5$ based on different covariance functions $k(\boldsymbol{r})$. For the first five samples, the applied length scales are $l_1 = 0.04$ in x– and $l_2 = 0.02$ in y–direction. The last sample is isotropic with $l = l_1 = 0.04$.

$p_{w_2}(w_2)$, which can be used to generate higher-dimensional samples from factorizing kernel functions.

To generate binary microstructures from a Gaussian process, a cutoff c_{cut} needs to be defined such that

$$\mathbb{1}_f(\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad g(\boldsymbol{x}) > c_{\text{cut}}, \\ 0 & \text{else}, \end{cases} \qquad \mathbb{1}_s(\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad g(\boldsymbol{x}) \leq c_{\text{cut}}, \\ 0 & \text{else}. \end{cases} \tag{2.19}$$

Given the Gaussian process variance $\sigma_g^2 = k(\mathbf{0})$, one can relate the cutoff parameter c_{cut} to the expected volume fractions $|\Omega_s|$, $|\Omega_f| = 1 - |\Omega_s|$ as

$$c_{\text{cut}} = \sigma_g F^{-1}(|\Omega_s|), \tag{2.20}$$

where $F^{-1}(\cdot)$ is the quantile function (inverse CDF) to the standard normal distribution. Figure 2.4 shows microstructural samples for different covariance functions $k(\boldsymbol{r})$ with expected volume fractions $|\Omega_f| = |\Omega_s| = 0.5$.

2.4.2 Generation of microstructures with fully connected pore space Ω_f

In the Stokes flow experiments that are presented in Chapter 6, it is important that the contemplated microstructural data exhibit fully connected pore/fluid spaces Ω_f because otherwise the pressure response $P(\boldsymbol{x})$ of the PDE Equation (2.5) may not be

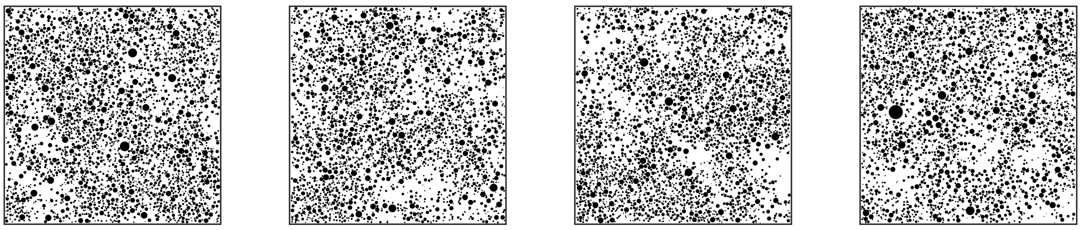


FIGURE 2.5: Typical microstructures of polydisperse spherical exclusions (black, solid phase $\Omega_s^{(n)}$) generated by the scheme described in Section 2.4.2. Figure taken from [87].

unique. We therefore use microstructures with non-overlapping spherical exclusions on the unit square domain $\Omega = [0,1]^2$.

Needless to say, microstructures based on spherical exclusions can at most be approximations to real-world microstructural data and as such, there is no natural process one could simulate to generate such kind of random media. The generation process for random binary media as used in the examples in Chapter 6 is therefore purely based on heuristics described in the following.

The simulation starts with the unit square domain $\Omega = [0,1]^2$ and successively substracts spherical subsets (i.e., exclusions) as long as they are not overlapping with previously substracted exclusions. The first number that needs to be drawn is thus the total number of exclusions $N_{\text{ex}}^{(n)}$ of microstructure n which is considered to be random as

$$N_{\text{ex}}^{(n)} \sim \text{round}\left[\text{Lognormal}(\mu_{ex}, \sigma_{ex}^2)\right]. \tag{2.21}$$

We then run over the exclusion index $i = 1 : N_{\text{ex}}^{(n)}$ of microstructure n and draw an exclusion center $\boldsymbol{x}_{\text{ex},i}^{(n)}$ and radius $r_{\text{ex, i}}^{(n)}$ according to

$$\boldsymbol{x}_{\text{ex},i}^{(n)} \sim \frac{1}{N_{\rho^{(n)}}} \rho_{\boldsymbol{x}_{\text{ex}}}^{(n)}(\boldsymbol{x}), \qquad \rho_{\boldsymbol{x}_{\text{ex}}}^{(n)}(\boldsymbol{x}) \sim S(GP(0, k_{\boldsymbol{x}}(\boldsymbol{x} - \boldsymbol{x}'))), \tag{2.22}$$

$$r_{\text{ex},i}^{(n)} \sim \text{Lognormal}(\mu_r^{(n)}(\boldsymbol{x}), \sigma_r^2), \tag{2.23}$$

where $\frac{1}{N_{\rho^{(n)}}} \rho_{\boldsymbol{x}_{\text{ex}}}^{(n)}(\boldsymbol{x})$ is a density drawn from a Gaussian process with stationary covariance $k_{\boldsymbol{x}}(\boldsymbol{x} - \boldsymbol{x}')$ warped by a logistic sigmoid $S(z) = 1/(1 + e^{-l_s z})$ of length scale l_s and $N_{\rho^{(n)}} = \int \rho_{\boldsymbol{x}_{\text{ex}}}^{(n)}(\boldsymbol{x})\boldsymbol{x}$ is the normalization constant. The location dependent expected exclusion radius $\mu_r^{(n)}(\boldsymbol{x})$ is also drawn from a Gaussian process

$$\mu_r^{(n)}(\boldsymbol{x}) \sim GP(0, k_r(\boldsymbol{x} - \boldsymbol{x}')) \tag{2.24}$$

with stationary covariance $k_r(\boldsymbol{x} - \boldsymbol{x}')$.

Given an exclusion center $\boldsymbol{x}_{\text{ex},i}^{(n)}$ and radius $r_{\text{ex},i}^{(n)}$, it is checked if the exclusion i overlaps with a previously inserted exclusion $j < i$. If no exclusion j is overlapping with exclusion i, exclusion i is inserted and appended to the list of exclusions $\boldsymbol{\lambda}_f^{(n)}$. If an overlapping exclusion is found, a new exclusion center/radius pair $\boldsymbol{x}_{\mathrm{ex},i}^{(n)}, r_{\mathrm{ex},i}^{(n)}$ is drawn until the insertion is successful[6]. Additionally, to avoid vigorous spikes in the Stokes flow pressure and velocity fields $P(\boldsymbol{x})$ and $\boldsymbol{V}(\boldsymbol{x})$ due to boundary conditions, exclusions are also rejected if they are on or too close to the domain boundary $\partial\Omega$. Some typical microstructures generated according to the above scheme are depicted in Figure 2.5.

[6]Of course, the described rejection procedure distorts the exclusion distribution from the distribution defined by Equation (2.22)–(2.24).

Chapter 3

Stochastic partial differential equations and the finite element method – numerical solution of Stokes and Darcy flow

The main purpose of this work is the development of cheap surrogates for numerical solvers of *partial differential equations* (PDEs) oneying uncertain input parameters such as PDE coefficients, boundary conditions or domain geometry – leading to what is known as *stochastic partial differential equations* (SPDEs), which are introduced in the sequel.

Let $(\Lambda, \mathcal{S}, P)$ a probability space with sample space Λ, σ-algebra $\mathcal{S}$ and probability measure $P : \mathcal{S} \mapsto [0, 1]$. Let $\Omega \subset \mathbb{R}^d$ be a d-dimensional bounded domain with boundary $\partial\Omega$. A linear stochastic partial differential equation can generally be written as

$$\begin{aligned} \mathcal{A}(\boldsymbol{x}, \lambda)u(\boldsymbol{x}, \lambda) &= s(\boldsymbol{x}, \lambda) \quad \text{for} \quad && \boldsymbol{x} \in \Omega(\lambda), \\ \mathcal{B}(u(\boldsymbol{x}, \lambda), \lambda) &= 0 \quad \text{for} \quad && \boldsymbol{x} \in \partial\Omega(\lambda), \end{aligned} \tag{3.1}$$

where $\mathcal{A}$ is a linear stochastic differential operator, $\boldsymbol{x} \in \Omega(\lambda)$, $\lambda \in \Lambda$ are elements of the physical and stochastic space, $u(\boldsymbol{x}, \lambda) \in \mathcal{H}$ is the solution field out of a Hilbert space $\mathcal{H}$ defined over the domain $\Omega(\lambda)$, $s(\boldsymbol{x}, \lambda)$ is a source term and $\mathcal{B}(u(\boldsymbol{x}, \lambda), \lambda)$ is a boundary condition operator. While the physical space Ω is assumed to have a finite number of dimensions, i.e., $\dim(\boldsymbol{x}) = d < \infty$, the stochastic input $\lambda \in \Lambda$ may in principle consist of infinite-dimensional random fields.

As apparent from Equation (3.1), uncertainty modeled by the stochastic input λ may be present in the differential operator $\mathcal{A}$ (e.g., as a random coefficient), in the source term s, in the boundary conditions $\mathcal{B}$ or even in the domain geometry Ω leading to a stochastic PDE response field $u(\boldsymbol{x}, \lambda)$. For instance, in the context of Darcy-type flow problems (see Section 2.2), a random coefficient in the differential operator $\mathcal{A}$ could model a random permeability field $\boldsymbol{K}(\boldsymbol{x}, \lambda)$, a random right hand side $s(\boldsymbol{x}, \lambda)$ would model a stochastic source field and random boundary conditions $\mathcal{B}(u(\boldsymbol{x}, \lambda), \lambda)$ could model randomness in the in- and outflow to the domain Ω. Moreover, as discussed in Section 2.1, uncertainty may even be present in the geometry of the domain $\Omega(\lambda)$.

Solving an SPDE means being capable to do inference w.r.t. the response statistics, i.e., to compute expected values

$$\langle f(u(\boldsymbol{x}))\rangle_{p(u)} = \int f(u(\boldsymbol{x}))p(u(\boldsymbol{x}))du(\boldsymbol{x}) = \int f(u(\boldsymbol{x}))\delta(u(\boldsymbol{x}) - u(\boldsymbol{x}, \lambda))p(\lambda)d\lambda \tag{3.2}$$

for any generic function $f(u(\boldsymbol{x}))$. Obviously, Equation (3.2) can be resolved to arbitrary accuracy using Monte Carlo methods as introduced in Section 4.3.3, making it the reference method for the solution of SPDEs. However, more often than not, the (deterministic) PDE solution $u(\boldsymbol{x}, \lambda)$ is not accessible in closed form and numerical methods need to be applied in order to get at least approximate solutions $\hat{u}_f(\boldsymbol{x}, \lambda) \approx u(\boldsymbol{x}, \lambda)$. One of the most prominent approaches is the *finite element method* (FEM), which is introduced in Section 3.1.

The biggest brake shoe in using numerical solvers combined with Monte Carlo to solve SPDEs is that these solvers typically require a high amount of computational resources for a single evaluation $\hat{u}_f(\boldsymbol{x}, \lambda^{(n)})$, whereas at the same time, Monte Carlo requires many such forward solves in order to produce convergent statistics.

This is the main incentive why in the last decades, many methods have been developed to solve Equation (3.2) with as few as possible forward iterations $\hat{u}_f^{(n)}(\boldsymbol{x}) = \hat{u}_f(\boldsymbol{x}, \lambda^{(n)})$ but simultaneously providing maximally accurate estimates of $\langle f(u)\rangle_{p(u)}$. Many of these methods are based on replacing the expensive numerical simulation $\hat{u}_f(\boldsymbol{x}, \lambda)$ by a cheaper, yet inaccurate model $\hat{u}_c(\boldsymbol{x}, \lambda)$ that allows to reconstruct $\hat{u}_f(\boldsymbol{x}, \lambda)$ and therefore to approximate Equation (3.2) at a fraction of the original cost. This framework is known as *surrogate*- or *metamodeling* and some of the most important surrogate modeling approaches that have been used for the solution of SPDEs are presented in Section 4.5.

3.1 Numerical solution of PDEs: the finite element method

The finite element method (FEM) [129–131] is a numerical method to find approximate solutions to boundary value problems of PDEs. After recasting the PDE into a *weak* or *variational* form, the approximate solution is searched in a finite-dimensional Hilbert space $\mathbb{V}_{FE} = \text{span}\left(\psi_1(\boldsymbol{x}), \ldots, \psi_{N_{dof}}(\boldsymbol{x})\right)$ spanned by N_{dof} localized *shape functions* $\psi_k(\boldsymbol{x})$. The result is a (typically large) set of algebraic equations which can be solved given sufficient computational resources.

The FEM approach is discussed paradigmatically in the following for stationary Stokes and Darcy flow, the only two PDEs that occur in this work.

3.1.1 Darcy flow simulation

The constitutive law for Darcy flow defined by Equation (2.6) can be rephrased into the linear elliptic PDE

$$\nabla_{\boldsymbol{x}} \cdot (\boldsymbol{K}\nabla_{\boldsymbol{x}}\bar{P}) = 0 \qquad \text{for} \quad \boldsymbol{x} \in \Omega, \tag{3.3a}$$

$$-(\boldsymbol{K}\nabla_{\boldsymbol{x}}\bar{P}) \cdot \boldsymbol{n} = \boldsymbol{V}_{bc} \cdot \boldsymbol{n} \qquad \text{for} \quad \boldsymbol{x} \in \Gamma_V, \tag{3.3b}$$

$$\bar{P} = P_{bc} \qquad \text{for} \quad \boldsymbol{x} \in \Gamma_P, \tag{3.3c}$$

which is called the *strong form* of the boundary value problem with solution $\bar{P} \in \mathbb{V}$ out of the solution space $\mathbb{V}$.

Equivalently, one may write Equation (3.3a) and (3.3b) as

$$\int_\Omega w(\boldsymbol{x}) \left(\nabla_{\boldsymbol{x}} \cdot (\boldsymbol{K}\nabla_{\boldsymbol{x}}\bar{P})\right) d\Omega = 0 \qquad \forall w \in \mathbb{W}, \tag{3.4a}$$

$$\int_{\Gamma_V} w(\boldsymbol{x}) \left(\boldsymbol{K}\nabla_{\boldsymbol{x}}\bar{P} + \boldsymbol{V}_{bc}\right) \cdot \boldsymbol{n} d\Gamma_V = 0 \qquad \forall w \in \mathbb{W}, \tag{3.4b}$$

where $w \in \mathbb{W}$ is an arbitrary *weight function* out of the *variation space* $\mathbb{W}$. Equation (3.4) is called the *weak form* that corresponds to the strong form Equation (3.3) in the sense that if $\bar{P}$ is a solution to Equation (3.3), it is also a solution to Equation (3.4) and vice versa. However, the weak form sets less strong conditions on the smoothness of the solution function $\bar{P}(\boldsymbol{x})$ and gives a foundation for the construction of approximate solutions.

It is generally desirable to have the order of derivatives as low as possible in the weak form, which is why we integrate Equation (3.4a) by parts to obtain

$$\int_{\partial\Omega} w(\boldsymbol{x})(\boldsymbol{K}\nabla_{\boldsymbol{x}}\bar{P}) \cdot \boldsymbol{n} d\Gamma - \int_\Omega \nabla_{\boldsymbol{x}}(w(\boldsymbol{x})) \cdot \boldsymbol{K}\nabla_{\boldsymbol{x}}(\bar{P}) d\Omega = 0. \tag{3.5}$$

The finite element method discretizes the above equation by the introduction of finite-dimensional function spaces $\mathbb{V}_{FE} \subset \mathbb{V}$ and $\mathbb{W}_{FE} \subset \mathbb{W}$ for both the solution $\bar{P}$ and the weight function w.

In the Bubnov-Galerkin method, these function spaces are chosen such that

$$\bar{P}_{FE}(\boldsymbol{x}) = u(\boldsymbol{x}) + \bar{P}_{bc}(\boldsymbol{x}), \tag{3.6}$$

where P_{FE} is the approximate finite element solution, $u(\boldsymbol{x}), w(\boldsymbol{x}) \in \mathbb{W}_{FE}$ and P_{bc} is a function that satisfies the essential boundary condition Equation (3.3c) on Γ_P. The crucial point of Equation (3.6) is that $u(\boldsymbol{x})$ and $w(\boldsymbol{x})$ are composed of the same basis functions. Note that for the essential boundary condition to hold, it is imperative that $u(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \Gamma_P$, which we enforce by choosing the function space $\mathbb{W}_{FE} = \text{span}\left(\psi_1(\boldsymbol{x}), \dots, \psi_{N_{dof}}(\boldsymbol{x})\right)$ s.t. this condition is fulfilled for all $w \in \mathbb{W}_{FE}$. This is achieved by requiring that for all shape functions ψ_i, it is claimed that

$$\psi_i(\boldsymbol{x}) = 0 \qquad \text{for} \quad \boldsymbol{x} \in \Gamma_P \tag{3.7}$$

which changes the integration domain of the first term in Equation (3.5) to give

$$-\int_{\Gamma_V} w(\boldsymbol{x}) \boldsymbol{V}_{bc}(\boldsymbol{x}) \cdot \boldsymbol{n} d\Gamma - \int_{\Omega} \nabla_{\boldsymbol{x}}(w(\boldsymbol{x})) \cdot \boldsymbol{K} \nabla_{\boldsymbol{x}}(\bar{P}) d\Omega = 0. \tag{3.8}$$

Plugging in the expansions

$$u(\boldsymbol{x}) = \sum_{i=1}^{N_{dof}} u_i \psi_i(\boldsymbol{x}), \qquad w(\boldsymbol{x}) = \sum_{i=1}^{N_{dof}} w_i \psi_i(\boldsymbol{x}). \tag{3.9}$$

of both the solution and weight functions $u, w \in \mathbb{W}_{FE}$ yields an equation with N_{dof} u_i's and w_i's, respectively. Given that the weak form Equation (3.4) needs to hold for *any* choice of weight function w, it must also hold for all possible choices of w_i's. This property can be used to equate coefficients in the w_i's, resulting in N_{dof} equations of N_{dof} solution coefficients u_i. In cases where the solution function $\bar{P}(\boldsymbol{x})$ (or equivalently, $u(\boldsymbol{x})$) appears only linearly in the PDE (which is the case in Darcy flow), the equation system is linear and can be written in matrix form as

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{F}, \tag{3.10}$$

where $\boldsymbol{A}$ is called the *stiffness matrix*, $\boldsymbol{u}$ the *solution vector* and $\boldsymbol{F}$ the *right hand side* or *force vector*. The assembly of $\boldsymbol{A}$ and $\boldsymbol{F}$ requires the solution of the integrals in Equation (3.8) which may be performed by efficient numerical schemes such as Gaussian quadrature.

Typically, the shape functions $\psi_i(\boldsymbol{x})$ are chosen to be localized, i.e.,

$$\psi_i(\boldsymbol{x}) \neq 0 \quad \text{for} \quad \boldsymbol{x} \in \Omega_k, \qquad \psi_i(\boldsymbol{x}) = 0 \quad \text{else}, \tag{3.11}$$

where

$$\Omega = \bigcup_{e=1}^{N_{el}} \Omega_e \qquad \Omega_e \cap \Omega_l = \emptyset \quad \forall e \neq l \tag{3.12}$$

which generally leads to sparse stiffness matrices $\boldsymbol{A}$ and is therefore computationally beneficial.

For the examples discussed in Chapter 6, we use square elements Ω_e and bilinear shape functions of the form

$$\psi_i(\boldsymbol{x}) = a_i^{(0)} + a_i^{(1)} x_1 + a_i^{(2)} x_2 + a_i^{(12)} x_1 x_2 \tag{3.13}$$

where the coefficients $a_i^{(0)}, a_i^{(1)}, a_i^{(2)}, a_i^{(12)}$ are chosen such that $\psi_i(\boldsymbol{x}_j) = \delta_{ij}$ with grid vertex $\boldsymbol{x}_j$. In such a way, it is ensured that $\bar{P}_{FE}(\boldsymbol{x}_j) = u_j$.

Assuming binary materials $\boldsymbol{K}(\boldsymbol{x}) = \boldsymbol{K}_s \mathbb{1}_s(\boldsymbol{x}) + \boldsymbol{K}_f \mathbb{1}_f(\boldsymbol{x})$ as defined in Equation (2.7) where the spatial shape of the (complementary) indicator functions $\mathbb{1}_f(\boldsymbol{x}) = \mathbb{1}_f(\boldsymbol{x}; \boldsymbol{\lambda})$, $\mathbb{1}_s(\boldsymbol{x}) = \mathbb{1}_s(\boldsymbol{x}; \boldsymbol{\lambda})$ is controlled by a random parameter $\boldsymbol{\lambda}$, the Darcy flow equations of motion Equation (3.3) turn into a stochastic PDE and the numerical solution vector

$$\boldsymbol{u}(\boldsymbol{\lambda}) = \boldsymbol{A}^{-1}(\boldsymbol{\lambda}) \boldsymbol{F}(\boldsymbol{\lambda}) \tag{3.14}$$

is dependent on the stochastic input vector $\boldsymbol{\lambda}$. A major component of this work is to replace the above equation by a cheap to evaluate surrogate in Chapter 5 for the purpose of uncertainty propagation [103, 104].

Gradients with respect to the random parameters λ

It is often the case that one is not only interested in the (discretized) solution $\boldsymbol{u}(\boldsymbol{\lambda})$, but also in a quantity of interest $G(\boldsymbol{u}(\boldsymbol{\lambda}))$ and its gradient $\nabla_{\boldsymbol{\lambda}} G(\boldsymbol{u}(\boldsymbol{\lambda}))$. This can be achieved by adding a 0 like

$$G(\boldsymbol{u}(\boldsymbol{\lambda})) = G(\boldsymbol{u}(\boldsymbol{\lambda})) - \xi_i \left(A_{ij} u_j - F_i \right) \tag{3.15}$$

where $\boldsymbol{\xi}$ is an arbitrary constant vector. The derivative w.r.t. λ_k then becomes

$$\begin{aligned} \frac{\partial G}{\partial \lambda_k} &= \frac{\partial G}{\partial u_j} \frac{\partial u_j}{\partial \lambda_k} - \xi_i \left(\frac{\partial A_{ij}}{\partial \lambda_k} u_j + A_{ij} \frac{\partial u_j}{\partial \lambda_k} - \frac{\partial F_i}{\partial \lambda_k} \right) \\ &= \left(\frac{\partial G}{\partial u_j} - \xi_i A_{ij} \right) \frac{\partial u_j}{\partial \lambda_k} - \xi_i \left(\frac{\partial A_{ij}}{\partial \lambda_k} u_j - \frac{\partial F_i}{\partial \lambda_k} \right). \end{aligned} \tag{3.16}$$

If $\boldsymbol{\xi}$ is chosen such that the first term vanishes, i.e., if

$$\boldsymbol{\xi} = \boldsymbol{A}^{-T} \nabla_{\boldsymbol{u}} G, \tag{3.17}$$

then

$$\frac{\partial G}{\partial \lambda_k} = \boldsymbol{\xi}^T \left(\frac{\partial \boldsymbol{F}}{\partial \lambda_k} - \frac{\partial \boldsymbol{A}}{\lambda_k} \boldsymbol{u} \right), \tag{3.18}$$

which allows to efficiently compute the gradient $\nabla_{\boldsymbol{\lambda}} G$. Equation (3.17) is called the *adjoint problem* and its solution is of similar cost as a forward solve $\boldsymbol{u} = \boldsymbol{A}^{-1} \boldsymbol{F}$.

3.1.2 Stokes flow simulation

Given the equations of motion for Stokes flow (see Equation (2.5))

$$\begin{aligned} \nabla_{\boldsymbol{x}} P - \nabla_{\boldsymbol{x}}^2 \boldsymbol{V} &= \boldsymbol{f} \qquad && \text{for} \quad \boldsymbol{x} \in \Omega_f, \\ \nabla_{\boldsymbol{x}} \cdot \boldsymbol{V} &= 0 \qquad && \text{for} \quad \boldsymbol{x} \in \Omega_f, \end{aligned}$$

we can establish the weak form

$$\int_{\Omega_f} \boldsymbol{U} \cdot \left(\nabla_{\boldsymbol{x}} P - \nabla_{\boldsymbol{x}}^2 \boldsymbol{V}\right) d\Omega_f = \int_{\Omega_f} \boldsymbol{U} \cdot \boldsymbol{f} d\Omega_f \qquad \forall \boldsymbol{U} \in \mathbb{W}^{\boldsymbol{U}}, \tag{3.19}$$

$$\int_{\Omega_f} (\nabla_{\boldsymbol{x}} \cdot \boldsymbol{V}) w d\Omega_f = 0 \qquad \forall w \in \mathbb{W}^{w}, \tag{3.20}$$

where $\boldsymbol{U}$ and w are arbitrary weight functions out of the weight function spaces $\mathbb{W}^{\boldsymbol{U}}$, $\mathbb{W}^{w}$ and $\dim(\boldsymbol{U}) = \dim(\boldsymbol{V})$. Again, to reduce the order of derivatives, one may integrate the first equation by parts to get

$$\int_{\partial\Omega_f} \left(\left(P\boldsymbol{I} - [\nabla_{\boldsymbol{x}} \boldsymbol{V}]\right) \boldsymbol{U}\right) \cdot \boldsymbol{n} d\Gamma + \int_{\Omega_f} [\nabla_{\boldsymbol{x}} \boldsymbol{U}] : [\nabla_{\boldsymbol{x}} \boldsymbol{V}] - (\nabla_{\boldsymbol{x}} \cdot \boldsymbol{U}) P d\Omega_f = \int_{\Omega_f} \boldsymbol{U} \cdot \boldsymbol{f} d\Omega_f, \tag{3.21}$$

where we use the definition that $[\nabla_{\boldsymbol{x}} \boldsymbol{V}]_{ij} = \frac{\partial V_j}{\partial x_i}$. From Equation (3.21), it is apparent that as boundary conditions on $\Gamma = \partial\Omega_f$, one either needs to specify $\boldsymbol{V} = \boldsymbol{V}_{bc}$ (where the weight function is chosen to be $\boldsymbol{U} = \boldsymbol{0}$) or the surface tractions $\boldsymbol{t} = (\nabla_{\boldsymbol{x}} \boldsymbol{V} - P\boldsymbol{I})$.

Again, the Bubnov-Galerkin approach is followed and the functions $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{W}_{FE}^{\boldsymbol{U}}$ and $P, w \in \mathbb{W}_F E^{w}$ are approximated by elements of identical finite-dimensional Hilbert spaces $\mathbb{W}_{FE}^{\boldsymbol{U}}, \mathbb{W}_{FE}^{w}$. However, not all choices of shape functions spanning the mixed finite element space $\mathbb{W}_{FE}^{\boldsymbol{U}} \times \mathbb{W}_{FE}^{w}$ are possible: the Ladyzhenskaya – Babuška--Brezzi compatibility condition (see, e.g., [132]) needs to be satisfied to ensure stability. Popular methods are, e.g., the Crouzeix-Raviart element [133], the MINI element [134], or the Taylor - Hood method [135] which is applied in this work to carry out stationary Stokes flow simulations.

The Taylor-Hood approach ensures stability by choosing the basis functions of $\mathbb{W}_{FE}^{\boldsymbol{U}}$ to be piecewise differentiable polynomials C^0 of order q and the basis functions of $\mathbb{W}_{FE}^{w}$ to be piecewise differentiable polynomials C^0 of order $q-1$, where we use $q = 2$. Clearly, the shape functions are again localized and the discretization of Ω_f is done with an irregular triangular mesh found using Delaunay triangulation [136]. Both mesh generation and the solution of the linear finite element system of equations are performed using the FEniCS finite element software package [137, 138].

Chapter 4

Elements of probability theory and uncertainty quantification

The purpose of this work is to develop a cheap-to-evaluate, yet sufficiently accurate class of surrogate models for the solution of stochastic partial differential equations. Although the proposed framework is widely applicable, we focus on the solution of flow through random porous media problems (Chapter 2) as the primary example.

By experiment, the detailed topology of such media, described by the vector $\boldsymbol{\lambda}$, can typically not be resolved in full detail. Moreover, in many engineering problems, it is not even required to know the exact solutions (i.e., pressure and velocity fields $\boldsymbol{u}^{(n)} = \left(\begin{smallmatrix} P^{(n)} \\ \boldsymbol{V}^{(n)} \end{smallmatrix}\right)$) for a certain set of microstructures $\left\{\boldsymbol{\lambda}^{(n)}\right\}_{n=1}^{N}$, but the *distribution* of solutions $p(\boldsymbol{u}) = p(P, \boldsymbol{V})$ given a probabilistic description of microstructures $p(\boldsymbol{\lambda})$. This is known as the *uncertainty propagation* (UP) *problem* presented in Section 4.1. As we will see in the experiments in Chapter 6, UP problems can be greatly accelerated assuming fast, but slightly inaccurate predictive models $p_{\text{pred}}(\boldsymbol{u}|\boldsymbol{\lambda}, \mathcal{D})$ based on a dataset $\left\{\boldsymbol{\lambda}^{(n)}, \boldsymbol{u}^{(n)}\right\}_{n=1}^{N}$ approximating the true forward model $\boldsymbol{u}(\boldsymbol{\lambda})$ without large deviations from the true posterior density $p(\boldsymbol{u})$.

The construction of surrogate models is typically a data-driven problem, where the model architecture is given as a stencil and model parameters are adapted such that the given data look most plausible. This is, in a manner of speaking, a (supervised) *machine learning problem*, which is what we introduce in Section 4.2 including all techniques that are needed for the model proposed in Chapter 5. Also, all the approximate inference methods required for understanding of Chapter 5 are presented in Section 4.3.

Machine learning models are trained by optimization of the model parameters. A popular class of optimization algorithms in machine learning is *stochastic gradient ascent* (SGA), which is used later in this work and therefore introduced in Section 4.4.

Finally, Section 4.5 introduces some common types of surrogate models that have been applied to stochastic PDEs and are therefore competing with the proposed framework. Pros and cons of each method are mentioned as a further motivation for the model introduced in Chapter 5.

4.1 The uncertainty propagation problem

Computer codes, for example the numerical solution of Stokes or Darcy flow in Equations (2.5), (2.6) by the methods discussed in Section 3.1 can be viewed as a function

$$\hat{\boldsymbol{u}} : \Lambda \mapsto U, \tag{4.1}$$

i.e., supplying an input $\boldsymbol{\lambda} \in \Lambda$ to the model will yield the response $\boldsymbol{u} = \hat{\boldsymbol{u}}(\boldsymbol{\lambda}), \boldsymbol{u} \in U$. In general, the function $\hat{\boldsymbol{u}}(\boldsymbol{\lambda})$ is not given in closed form but can be evaluated pointwise by running the corresponding computer code.

Uncertainty propagation (UP) addresses the question of how the model output $\boldsymbol{u}$ is distributed given a distribution of inputs $\boldsymbol{\lambda}$. This can formally be written as

$$p(\boldsymbol{u}) = \int p(\boldsymbol{u}|\boldsymbol{\lambda})p(\boldsymbol{\lambda})d\boldsymbol{\lambda} = \int \delta(\boldsymbol{u} - \hat{\boldsymbol{u}}(\boldsymbol{\lambda}))p(\boldsymbol{\lambda})d\boldsymbol{\lambda}. \tag{4.2}$$

Solving the UP problem in closed form is seldom possible, but also often unnecessary: in many cases, it is sufficient to know *expected values* only, such as, for example, the *mean*

$$\langle \boldsymbol{u} \rangle = \int \boldsymbol{u} p(\boldsymbol{u})d\boldsymbol{u} = \int \boldsymbol{u}\delta(\boldsymbol{u} - \hat{\boldsymbol{u}}(\boldsymbol{\lambda}))p(\boldsymbol{\lambda})d\boldsymbol{\lambda}d\boldsymbol{u} = \int \hat{\boldsymbol{u}}(\boldsymbol{\lambda})p(\boldsymbol{\lambda})d\boldsymbol{\lambda} \tag{4.3}$$

which may be estimated with, e.g., *Monte Carlo* (MC, see Section 4.3.3) as

$$\langle \boldsymbol{u} \rangle \approx \frac{1}{N}\sum_{n=1}^{N} \hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(n)}), \qquad \boldsymbol{\lambda}^{(n)} \sim p(\boldsymbol{\lambda}^{(n)}). \tag{4.4}$$

It is well known that the MC error diminishes as $1/\sqrt{N}$ – this means that to obtain one digit higher accuracy in $\langle \boldsymbol{u} \rangle$, the number of samples N need to be increased by a factor of 100.

However, many computer simulations of practical interest require runtimes from hours up to several weeks to finish, so that only a few tens or hundreds of samples $\boldsymbol{\lambda}^{(n)}, \hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(n)})$ can be produced, which may be insufficient for accurate MC estimates even if direct sampling from $p(\boldsymbol{\lambda})$ is possible. It is thus of primary interest to squeeze as much information as possible from the dataset $\mathcal{D} = \left\{\boldsymbol{\lambda}^{(n)}, \hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(n)})\right\}_{n=1}^{N}$.

A vital concept for this is *surrogate*- or *meta-modeling*. Based on the data $\mathcal{D}$, a surrogate model attempts to emulate the response surface of the forward model $\hat{\boldsymbol{u}}(\boldsymbol{\lambda})$, at a typically much lower cost than evaluating $\hat{\boldsymbol{u}}(\boldsymbol{\lambda})$ at the expense of accuracy. A probabilistic surrogate in the form of a conditional density $p_{\text{pred}}(\boldsymbol{u}|\boldsymbol{\lambda},)$ is even capable to quantify the loss of accuracy. In UP, it can be directly plugged in to Equation (4.2) to give

$$p(\boldsymbol{u}) = \int p_{\text{pred}}(\boldsymbol{u}|\boldsymbol{\lambda})p(\boldsymbol{\lambda})d\boldsymbol{\lambda}, \tag{4.5}$$

so that the computation of expected values under $p(\boldsymbol{u})$ does not require to solve the forward model $\hat{\boldsymbol{u}}(\boldsymbol{\lambda})$ anymore.

The construction of a probabilistic surrogate $p_{\text{pred}}(\boldsymbol{u}|\boldsymbol{\lambda},)$ can be seen as a supervised learning problem, which is why we give an introduction to Bayesian statistics and machine learning in the subsequent section.

4.2 Machine learning and Bayesian statistics – an introduction

Machine learning is the science of statistical models and computer algorithms that can be used to solve a specific problem based on prior experience rather than hard-coded instructions. An incisive definition is given by T. M. Mitchell in [139]:

> *"A computer program is said to* ***learn*** *from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."*
> (Tom M. Mitchell, 1990)

A classic example is object recognition [64] in computer vision: The *task T* is to decide, based on pixel values of the digitized image, if a scenery contains objects like a cat, a car, an airplane, etc. Instead of explicitly programming decision rules (in the sense of, e.g., "a car has four wheels", "an airplane has two wings", etc.), a machine learning program uses a large set of previously analyzed images that have been labeled with, e.g., "contains a cat", "contains a car", etc. This set of previously analyzed images, the *training data*, is what is meant with *experience E* in the above definition.

The machine learning program for the above task can be seen as a mathematical function $f_{\boldsymbol{\theta}} : \boldsymbol{\lambda} \mapsto \boldsymbol{u}$ that maps the input $\boldsymbol{\lambda}$, i.e., the pixel values of an image, to the output $\boldsymbol{u}$, which is the class an image may belong to (e.g. "cat", "car", ...). In the stage of the analysis of the training data, called the *model training*, some free parameters $\boldsymbol{\theta}$ of the machine learning model $f_{\boldsymbol{\theta}}$ are adjusted such that a *loss function*, e.g., the missclassification rate of $f_{\boldsymbol{\theta}}$ on the training data, is minimized.

The *performance measure P* of the above definition could be the missclassification rate of $f_{\boldsymbol{\theta}}$ on a separate set of images not contained in the training data, the so-called *test dataset*. As stated above, on average, P should improve with increasing number of images in the training data.

The mathematical framework of machine learning is given by statistics and probability theory. In statistics, *frequentist* and *Bayesian* viewpoints to probability (see, e.g., [140]) differ in the way probability of a random parameter $\boldsymbol{\theta}$ is interpreted. While frequentists view the probability of an event as its relative frequency in the limit

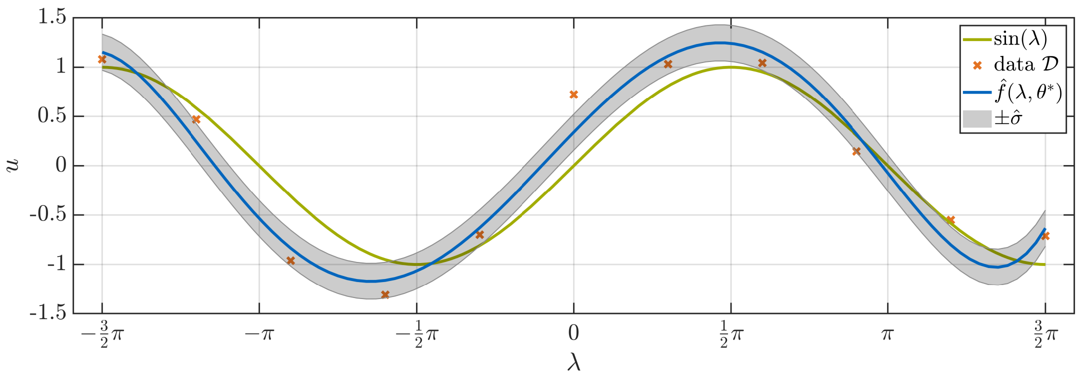


FIGURE 4.1: Linear regression example. The true function (green line) is $f(\lambda) = \sin(\lambda)$. The data points (orange dots) are generated by adding Gaussian noise with a standard deviation of $\sigma = 0.3$. The linear regression model (blue line) is the maximum likelihood (ML) solution based on a 4th order polynomial model $\hat{f}(\boldsymbol{\lambda}, \boldsymbol{\theta}^*) = \sum_{k=0}^{4} \hat{\theta}^*_{k+1} \lambda^k$.

of infinite trials, Bayesians interpret it as a subjective *degree of belief* that reflects the knowledge about the uncertain parameter $\boldsymbol{\theta}$. The difference in both interpretations may seem quite subtle and can create more confusion rather than being helpful for a deeper understanding. Nevertheless, concepts like *prior* and *posterior* probabilities appear more natural from a Bayesian perspective, which is thus the viewpoint adopted in this work.

4.2.1 Linear regression and maximum likelihood

Linear regression is one of the simplest possible tasks in the field of machine learning. It is therefore used here to explain some basic ideas and notions that will frequently reoccur in the rest of this thesis.

Figure 4.1 shows some data points (orange dots) $\mathcal{D} = \{\lambda_n, u_n\}_{n=1}^N$ generated from a random process

$$u_n = \sin(\lambda_n) + \sigma Z_n, \qquad Z_n \sim \mathcal{N}(0,1). \tag{4.6}$$

Without knowing the underlying process given by Equation (4.6), the goal is to find a function or, better yet, a probability distribution $p_{\text{pred}}(u|\lambda)$ that allows to *predict* the outcome u of the above process given a particular input λ.

Assuming Gaussian noise and that the data points are independent and identically distributed (i.i.d.), the *likelihood function* is

$$\mathcal{L}(\mathcal{D}|\hat{\boldsymbol{\theta}}, \hat{\sigma}^2) = \prod_{n=1}^{N} \mathcal{N}(u_n|\hat{f}(\lambda_n, \hat{\boldsymbol{\theta}}), \hat{\sigma}^2) \tag{4.7}$$

where $\hat{f}(\lambda, \hat{\boldsymbol{\theta}})$ is the model function depending on some parameters $\hat{\boldsymbol{\theta}}$ and $\hat{\sigma}^2$ is the model variance. The likelihood function gives the probability density to observe the data $\mathcal{D}$ given the model parameters $\boldsymbol{\theta} = \{\hat{\boldsymbol{\theta}}, \hat{\sigma}^2\}$. On the other hand, the likelihood function $\mathcal{L}(\mathcal{D}|\boldsymbol{\theta})$ can be viewed as a function of the model parameters $\boldsymbol{\theta}$ after some data $\mathcal{D}$ have been recorded. Given $\mathcal{D}$, it is a reasonable strategy to maximize the (log) likelihood function $\log \mathcal{L}(\mathcal{D}|\boldsymbol{\theta})$ w.r.t. the parameters $\boldsymbol{\theta}$

$$\begin{aligned}\hat{\boldsymbol{\theta}}^*, \hat{\sigma}^{2,*} &= \arg\max_{\hat{\boldsymbol{\theta}},\hat{\sigma}^2} \log \mathcal{L}(\mathcal{D}|\hat{\boldsymbol{\theta}}, \hat{\sigma}^2) \\ &= \arg\max_{\hat{\boldsymbol{\theta}},\hat{\sigma}^2} \left(-\frac{N}{2}\log\hat{\sigma}^2 - \frac{1}{2}\hat{\sigma}^{-2}\sum_{n=1}^{N}(u_n - \hat{f}(\lambda_n, \hat{\boldsymbol{\theta}}))^2\right)\end{aligned} \tag{4.8}$$

in order to *learn* of the underlying process Equation (4.6) and being able to reconstruct it with a *predictive distribution*

$$p_{\text{pred}}(u|\lambda, \hat{\boldsymbol{\theta}}^*, \hat{\sigma}^{2,*}) = \mathcal{N}(u|\hat{f}(\lambda, \hat{\boldsymbol{\theta}}^*), \hat{\sigma}^{2,*}). \tag{4.9}$$

Assuming a *linear model* of the form

$$\hat{f}(\lambda, \hat{\boldsymbol{\theta}}) = \sum_{k=1}^{K} \hat{\theta}_k \varphi_k(\lambda) \tag{4.10}$$

and setting the gradients of $\log \mathcal{L}$ w.r.t. the model parameters $\boldsymbol{\theta}, \hat{\sigma}^2$ to zero yields the ML[1] solution

$$\begin{aligned}0 &= \nabla_{\boldsymbol{\theta}}\left(-\frac{N}{2}\log\hat{\sigma}^2 - \frac{1}{2}\hat{\sigma}^{-2}\sum_{n=1}^{N}(u_n - \hat{f}(\lambda_n, \hat{\boldsymbol{\theta}}))^2\right) \\ &= \nabla_{\boldsymbol{\theta}}\left(-\frac{N}{2}\log\hat{\sigma}^2 - \frac{1}{2}\hat{\sigma}^{-2}(\boldsymbol{u} - \boldsymbol{\Phi}\hat{\boldsymbol{\theta}})^T(\boldsymbol{u} - \boldsymbol{\Phi}\hat{\boldsymbol{\theta}})\right), \\ \hat{\boldsymbol{\theta}}^* &= (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{u}, \qquad \hat{\sigma}^{2,*} = \frac{1}{N}(\boldsymbol{u} - \boldsymbol{\Phi}\hat{\boldsymbol{\theta}})^T(\boldsymbol{u} - \boldsymbol{\Phi}\hat{\boldsymbol{\theta}}),\end{aligned} \tag{4.11}$$

with the *design matrix* $\boldsymbol{\Phi}$, $\Phi_{nk} = \varphi_k(\lambda_n)$ and the solution vector $\boldsymbol{u} = \begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix}$. Figure 4.1 depicts the ground truth $\sin(\lambda)$ (green line) and the ML solution (blue line) of a model $\hat{f}(\lambda, \hat{\boldsymbol{\theta}}) = \sum_{k=0}^{4} \hat{\theta}_{k+1} x^k$ to the data (orange dots) given by Equation (4.6). The grey shaded areas represent the predictive uncertainty $\pm\hat{\sigma}$.

4.2.2 Bayesian regression and sparsity enforcing priors

Bayes' law

As already mentioned in the previous subsection, the likelihood function $\mathcal{L}(\mathcal{D}|\boldsymbol{\theta})$ is a valid probability distribution (i.e., it is strictly positive and normalized) w.r.t. the data $\mathcal{D}$. It is thus possible to apply *Bayes' law*[141]

$$\mathcal{L}(\mathcal{D}|\boldsymbol{\theta})p_0(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathcal{D})p(\mathcal{D}) = p(\boldsymbol{\theta}, \mathcal{D}) \tag{4.12}$$

[1] In the frequentist statistics literature, the maximum likelihood solution for a linear model with Gaussian noise is often called the *least squares solution*, as it minimizes the squared distance of $\hat{f}$ to the data.

which is a direct consequence of the product rule of probabilities and the definition of conditional distributions. In Equation (4.12), $p_0(\boldsymbol{\theta})$ is a *prior distribution*, $p(\boldsymbol{\theta}|\mathcal{D})$ is the *posterior distribution* and $p(\mathcal{D})$ is called the *evidence*.

It is worth to hold on for a moment and think about what Equation (4.12) really means: Instead of searching for a point estimate $\boldsymbol{\theta}^*$ that maximizes the likelihood function, an a priori probability distribution $p_0(\boldsymbol{\theta})$ is assigned over the model parameters $\boldsymbol{\theta}$, so that Bayes' law can be used to find a posterior distribution

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{\mathcal{L}(\mathcal{D}|\boldsymbol{\theta})p_0(\boldsymbol{\theta})}{p(\mathcal{D})} \propto \mathcal{L}(\mathcal{D}|\boldsymbol{\theta})p_0(\boldsymbol{\theta}) \tag{4.13}$$

that balances the a priori information encoded in $p_0(\boldsymbol{\theta})$ and the information from the data in $\mathcal{L}(\mathcal{D}|\boldsymbol{\theta})$ to reflect all the knowledge we have about the parameters $\boldsymbol{\theta}$ *after* the data $\mathcal{D}$ were seen. As the model evidence $p(\mathcal{D})$ does not depend on the parameters $\boldsymbol{\theta}$, it only plays the role of a normalization constant in Equation (4.13), but will become important in model selection tasks, see Section 4.2.3 and Section 6.2.10.

Given a model $\hat{f}(\lambda, \hat{\boldsymbol{\theta}})$ like, e.g., the one defined in Equation (4.10), the only thing left to specify is the prior $p_0(\boldsymbol{\theta})$. The prior distribution $p_0(\boldsymbol{\theta})$ should reflect all assumptions that can be made on $\boldsymbol{\theta}$ *before* any data have been recorded. For example, it is often possible to know a priori that certain parameters θ_i are strictly positive, $\theta_i > 0$, or limited to a certain interval, $\theta_i \in [a, b]$. Both can be realized by constructing a prior with $p_0(\boldsymbol{\theta}) = 0$ for $\theta_i \leq 0$ or $\theta_i \notin [a, b]$, respectively. Another possibility is to know a priori that, out of all parameters θ_i, only a few assume nonzero values. This can be achieved with *sparsity-enforcing priors*. The most popular ones are introduced in the following subsections.

Finally, given the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$, the predictive distribution can be constructed as

$$p_{\text{pred}}(u|\lambda) = \int p(u|\lambda, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta} \tag{4.14}$$

with $p(u|\lambda, \boldsymbol{\theta}) = \mathcal{N}(u|\hat{f}(\lambda, \hat{\boldsymbol{\theta}}), \hat{\sigma}^2)$ in the above example. Compared to the ML version in Equation (4.9), the above predictive distribution tends to be more accurate (dependent on the choice of the prior p_0, of course) as it also reflects uncertainties within the model parameters $\boldsymbol{\theta}$.

The Gaussian prior

Gaussian priors are actually not sparsity enforcing priors as they do not drive parameters to 0 exactly. However, it is the simplest possible prior in linear models with Gaussian noise and forms the basis of a couple of other priors discussed in later subsections. In the literature, this method is also known as L_2- or *Tikhonov regularization [142], ridge regression*[143] (statistics), *weight decay* [64] (deep learning) and several other names.

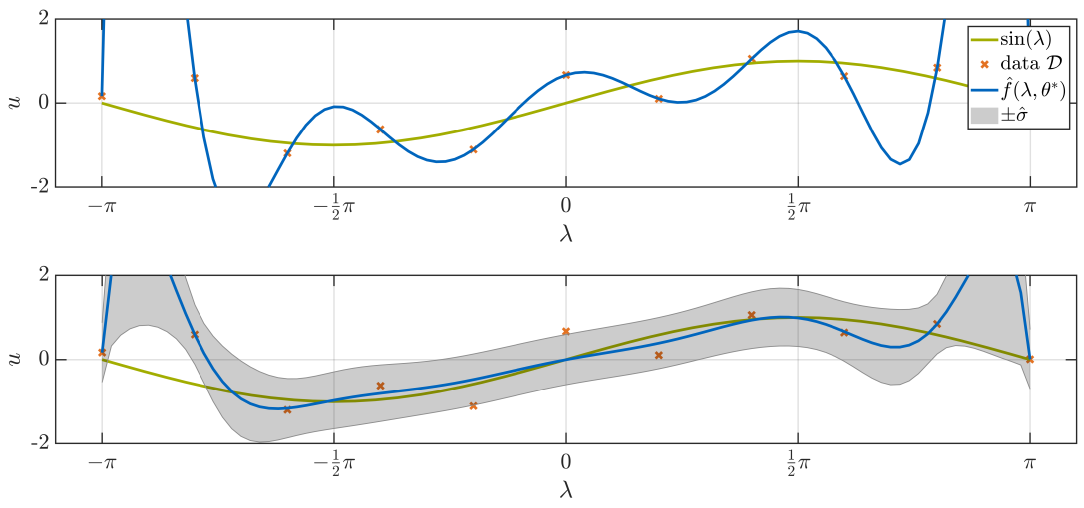


FIGURE 4.2: Comparison of ML linear regression (top) and a linear regression with a zero-mean Gaussian prior with precision $\gamma = 1$ on the model parameters $\hat{\boldsymbol{\theta}}$. The 11 data points (orange dots) are generated from the true function $f(\lambda) = \sin(\lambda)$ (green line) by adding Gaussian noise with standard deviation $\sigma = 0.5$. The model $\hat{f}$ is a 9th order polynomial $\hat{f}(\lambda) = \sum_{k=0}^{9} \hat{\theta}_{k+1}^{*} \lambda^k$, i.e., the ML regression passes the data exactly, but misses to fit the underlying function. One can observe that the regularizing effect of the Gaussian prior is an effective way to avoid overfitting. The likelihood variance $\hat{\sigma}^2$ for the Gaussian prior model was set to the variance of the data, $\hat{\sigma} = \sigma = 0.5$. and the error $\pm\sigma_{\text{pred}}$ (see Equation (4.20)) is represented with the grey shaded areas.

Assuming a Gaussian prior is equivalent to[2]

$$p_0(\hat{\boldsymbol{\theta}}) = \mathcal{N}(\hat{\boldsymbol{\theta}}|\mathbf{0}, \gamma^{-1}\boldsymbol{I}) \tag{4.15}$$

where $\boldsymbol{I}$ is the identity matrix and γ a precision parameter to be defined by the user. The log posterior $\log p(\hat{\boldsymbol{\theta}}|\mathcal{D}, \hat{\sigma}^2, \gamma)$ for a linear model as defined by Equation (4.10) then becomes

$$\log p(\hat{\boldsymbol{\theta}}|\mathcal{D}, \hat{\sigma}^2, \gamma) \propto -\frac{1}{2}\hat{\sigma}^{-2}(\boldsymbol{u} - \boldsymbol{\Phi}\hat{\boldsymbol{\theta}})^T(\boldsymbol{u} - \boldsymbol{\Phi}\hat{\boldsymbol{\theta}}) - \frac{1}{2}\gamma\hat{\boldsymbol{\theta}}^T\hat{\boldsymbol{\theta}}. \tag{4.16}$$

Similarly to the ML approach discussed in Section 4.2.1, a reasonable strategy is to maximize the log posterior w.r.t. $\hat{\boldsymbol{\theta}}$ to find the *maximum a posteriori* (MAP) estimate $\hat{\boldsymbol{\theta}}^*$. The log-posterior $\log p(\hat{\boldsymbol{\theta}}|\mathcal{D}, \hat{\sigma}^2, \gamma)$ is quadratic in $\hat{\boldsymbol{\theta}}$ and therefore a Gaussian as well. The posterior mean is thus unique and identical to the posterior mode

$$\hat{\boldsymbol{\theta}}^* = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \hat{\sigma}^2\gamma\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{u} \tag{4.17}$$

and the posterior covariance $\boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}}$ is

$$\boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}} = (\hat{\sigma}^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \gamma\boldsymbol{I})^{-1}. \tag{4.18}$$

[2]Obviously also Gaussians with non-zero mean can be applied as a prior distribution.

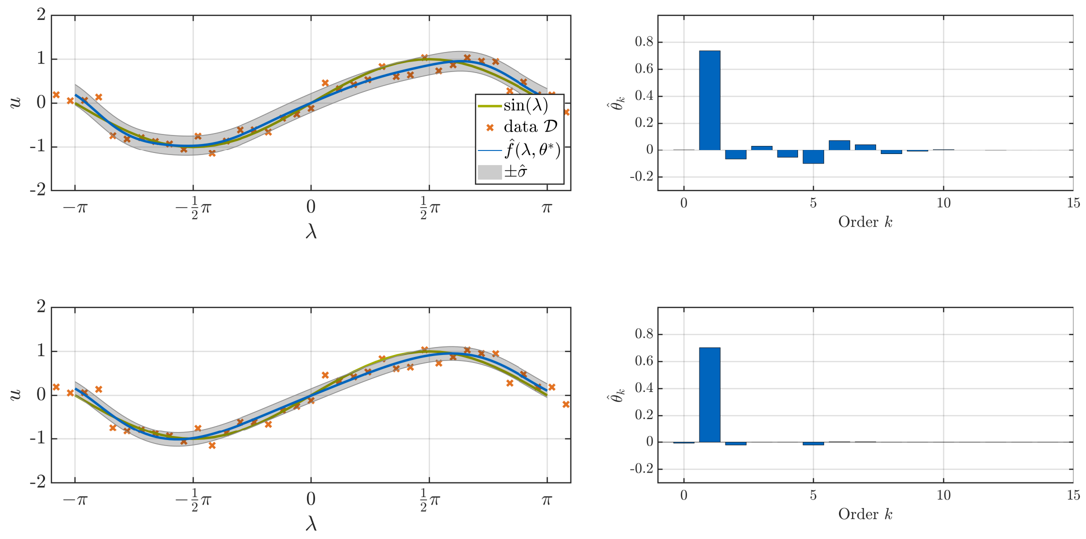


FIGURE 4.3: Comparison of a regression problem using a Gaussian (top) and a Laplacian prior (bottom) with precision parameter $\gamma = 10$ on the model parameters $\hat{\boldsymbol{\theta}}$. 51 data points (orange dots) are generated from the function $f(\lambda) = \sin(\lambda)$ (green line) by adding Gaussian noise with $\sigma = 0.2$. The model function is $f(\lambda) = \sum_{k=0}^{K} \hat{\theta}_k x^k$. The bar plots on the right show the posterior maximum $\hat{\boldsymbol{\theta}}^*$. It can be observed that $\hat{\boldsymbol{\theta}}^*$ is much sparser for the Laplacian prior.

Compared to the ML approach, the Gaussian prior favours smaller euclidean norms of $\boldsymbol{\theta}$ and is therefore regularizing the problem, i.e., it reduces model complexity and therefore avoids overfitting. Its effect can be observed in Figure 4.2.

The predictive distribution can be found by marginalizing the model parameters $\hat{\boldsymbol{\theta}}$ over the posterior

$$p(u|\lambda, \mathcal{D}, \hat{\sigma}^2, \gamma) = \int p(u|\lambda, \hat{\boldsymbol{\theta}}, \hat{\sigma}^2) p(\hat{\boldsymbol{\theta}}|\mathcal{D}, \hat{\sigma}^2, \gamma) d\hat{\boldsymbol{\theta}} = \mathcal{N}(u|\mu_{\text{pred}}(\lambda), \sigma^2_{\text{pred}}(\lambda)), \quad (4.19)$$

with the model basis functions $\boldsymbol{\varphi}(\lambda) = \begin{pmatrix} \varphi_0(\lambda) \\ \vdots \\ \varphi_K(\lambda) \end{pmatrix}$ and

$$\sigma^2_{\text{pred}}(\lambda) = \hat{\sigma}^2 + \boldsymbol{\varphi}^T(\lambda)\boldsymbol{\Sigma}_{\hat{\theta}}\boldsymbol{\varphi}(\lambda), \quad (4.20)$$

$$\mu_{\text{pred}}(\lambda) = \boldsymbol{\varphi}^T(\lambda)\hat{\boldsymbol{\theta}}^*. \quad (4.21)$$

Apart from the question how to find a suitable estimate for the likelihood variance $\hat{\sigma}$ (it depends on $\hat{\boldsymbol{\theta}}^*$ which itself depends on $\hat{\sigma}$), it is nontrivial to find a suitable value for the prior precision γ. This will question be answered in the discussion of other prior models in a later part of this section.

The Laplacian prior

Naturally, the effect of a Gaussian prior is *parameter shrinkage*, i.e., model parameters $\hat{\boldsymbol{\theta}}$ are shifted towards zero compared to only data-based (i.e., ML) estimates. As the regularization term $-\frac{1}{2}\gamma\hat{\boldsymbol{\theta}}^T\hat{\boldsymbol{\theta}} = -\frac{1}{2}\gamma\|\hat{\boldsymbol{\theta}}\|^2 = -\frac{1}{2}\gamma\sum_{k=0}^{K}\hat{\theta}_k^2$ is quadratic in $\hat{\boldsymbol{\theta}}$, the shrinkage is strong for large $\hat{\theta}_k$ and vanishes to 1st order for $\hat{\theta}_k \to 0$. To promote sparsity, i.e., $\hat{\theta}_k = 0$ for irrelevant predictors $\varphi_k(\lambda)$, this is counter-productive: there should be non-vanishing shrinkage for small components $\hat{\theta}_k \ll 1$ to push them to exactly 0. On the other side, important components (i.e., $\hat{\theta}_k$ large) should undergo smaller shrinkage to minimize bias introduced by the prior distribution.

One approach is to generalize to priors that lead to regularization terms of the form

$$\log p_0(\hat{\boldsymbol{\theta}}) \propto -\frac{1}{2}\gamma\|\hat{\boldsymbol{\theta}}\|_q^q = -\frac{1}{2}\gamma\sum_{k=0}^{K}|\hat{\theta}_k|^q, \tag{4.22}$$

where $\|\hat{\boldsymbol{\theta}}\|_q = \left(\sum_{k=0}^{K}|\hat{\theta}_k|^q\right)^{(1/q)}$ is the standard ℓ_q-norm of $\hat{\boldsymbol{\theta}}$. Indeed, the smaller the generalized norm[3] parameter q, the stronger the prior penalty for $\hat{\theta}_k \ll 1$ and the weaker for $\hat{\theta}_k \gg 1$.

For the extremal case, $q = 0$, the prior penalty $-\frac{1}{2}\gamma\sum_{k=0}^{K}|\hat{\theta}_k|^0$ would equal the number of nonzero components in $\hat{\boldsymbol{\theta}}$[4]. Finding a MAP estimate $\hat{\boldsymbol{\theta}}^*$ under a prior leading to ℓ_0 regularization is the so-called *best-subset selection* problem. The ℓ_0-norm is not convex and finding $\hat{\boldsymbol{\theta}}$ is therefore a combinatorially hard optimization problem [144]. In general, all ℓ_q-norms for $q < 1$ are non-convex functions, as can be seen, e.g., for the $\ell_{1/2}$-norm in the 3rd column of Figure 4.4.

In a linear model with Gaussian noise, the log-likelihood $\log\mathcal{L}(\mathcal{D}|\hat{\boldsymbol{\theta}},\hat{\sigma}^2)$ is a concave function in $\hat{\boldsymbol{\theta}}$, see Equation (4.11). The ML estimate for $\hat{\boldsymbol{\theta}}$ is therefore unique and can be found analytically. As the (nonnegative) sum of two concave functions is a concave function, the log posterior is concave if a concave prior is applied. It therefore makes sense to search for the concave log prior that has the best shrinking properties in the sense discussed above. This is exactly the ℓ_1 regularization corresponding to a Laplacian prior

$$p_0(\hat{\theta}_k) = \frac{\gamma}{2}\exp\{-\gamma|\hat{\theta}_k|\}. \tag{4.23}$$

As the log posterior $\log p(\hat{\boldsymbol{\theta}}|\mathcal{D})$ is concave, there is a unique MAP estimate $\hat{\boldsymbol{\theta}}^*$ and finding it is a concave optimization problem. In statistics literature, ℓ_1-regularization is commonly known as *Lasso regression* [145].

A persisting intricacy using a Laplacian prior (i.e., ℓ_1-regularization) is that this prior and hence the posterior is not a differentiable function, s.t. traditional (stochastic) gradient ascent methods are not convergent because gradients do not vanish at the optimum. However, several other model training methods exist, e.g., based on coordinate descent [146], expectation-maximization [147] (see Section 4.3.1), or least angle regression [148].

[3]For $q < 1$, the ℓ_q-norm does not fulfill the triangle inequality and therefore only defines a quasi-norm.

[4]Under the definition that $0^0 = 0$.

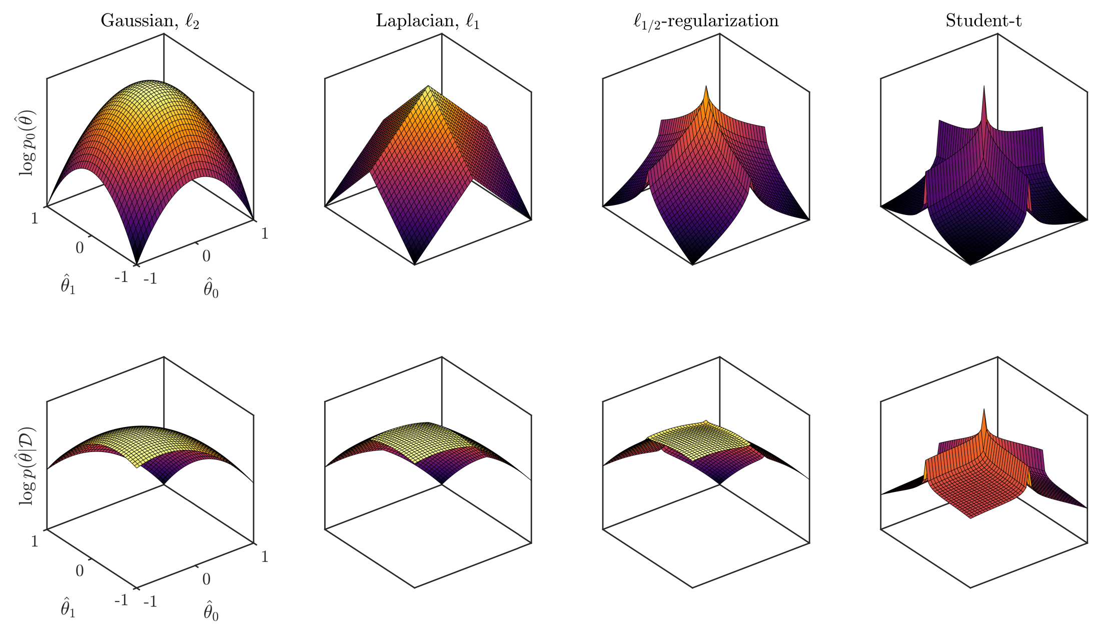


FIGURE 4.4: Top row: Different concave (Gaussian, Laplacian) and non-concave priors ($\ell_{1/2}$, Student-t).
Bottom row: The corresponding posteriors assuming a Gaussian likelihood of the form $\log \mathcal{L}(\mathcal{D}|\hat{\boldsymbol{\theta}}) \propto -\frac{3}{2}\|\hat{\boldsymbol{\theta}} - (-1,\ -1)^T\|^2$. The Student-t prior is constructed via $p_0(\hat{\theta}_k) = \int_0^\infty \mathcal{N}(\hat{\theta}_k|0, \gamma_k)\Gamma(\gamma_k|a_0, b_0)d\gamma_k$ where $\Gamma(\gamma_k|a_0, b_0)$ is a *Gamma* distribution and $a_0 = 0, b_0 = 10^{-6}$. It can be observed in the bottom row that non-concave log priors generate local optima in the posterior.

The Student-t prior

Although several heuristics exist [149], a central question in both Gaussian and Laplacian priors is how to assign reasonable values to the hyperparameter γ *a priori*. One strategy to accommodate the uncertainty in the hyperparameter γ is to consider it as a random variable and average w.r.t. another layer of prior distribution (i.e., a *hyperprior*). In order to retain analytical tractability, a conjugate *Gamma* hyperprior is applied,

$$\Gamma(\gamma|a_0, b_0) = \frac{b_0^{a_0}\gamma^{a_0-1}e^{-b_0\gamma}}{\Gamma(a_0)}. \tag{4.24}$$

where $\Gamma(a_0)$ is the gamma function[5] and $a_0, b_0 > 0$ are distribution parameters. Being able to get rid of the hyperprior γ, it is possible to generalize to the *automatic relevance determination* (ARD) prior [150, 151] and assign distinct hyperparameters γ_k for different parameters $\hat{\theta}_k$,

$$p(\hat{\boldsymbol{\theta}}|\gamma) = \prod_{k=0}^{K} \mathcal{N}(\hat{\theta}_k|0, \gamma_k^{-1}). \tag{4.25}$$

[5]Note that the symbol $\Gamma(a_0) = \int_0^\infty x^{a_0-1}e^{-x}dx$ with a single argument denotes the gamma function; the symbol $\Gamma(\gamma|a_0, b_0)$ denotes the *Gamma distribution* over γ with parameters a_0, b_0.

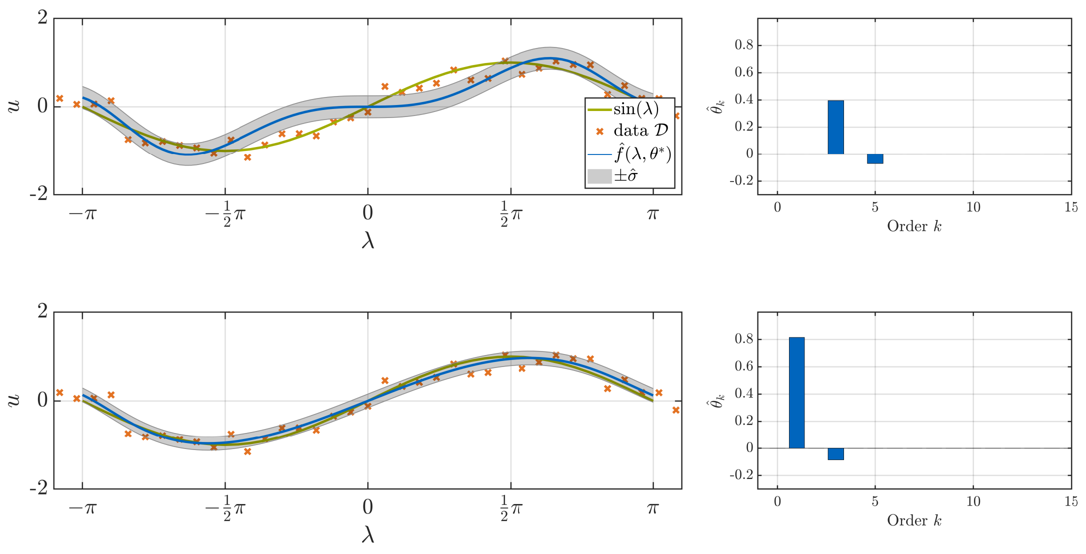


FIGURE 4.5: Regression problem using the Student-t prior with $a_0 = b_0 = 10^{-10}$ initialized at $\hat{\boldsymbol{\theta}} \sim \prod_k \mathcal{N}(\hat{\theta}_k | \mu = 0, \sigma^2 = 0.01)$ (top) and $\hat{\boldsymbol{\theta}} = (1, \dots, 1)^T$ using the same data as in Figure 4.3. Different MAP estimates $\hat{\boldsymbol{\theta}}^*$ are observed for different initializations – a symptom of multimodality.

Averaging over γ, a Student-t type prior $p_0(\hat{\boldsymbol{\theta}}) = \prod_k p_0(\hat{\theta}_k)$ with

$$p_0(\hat{\theta}_k) = \int \mathcal{N}(\hat{\theta}_k | 0, \gamma_k^{-1}) \Gamma(\gamma_k | a_0, b_0) d\gamma = \frac{b_0^{a_0} \Gamma(a_0 + \frac{1}{2})}{\sqrt{2\pi}\Gamma(a_0)} \left(b_0 + \frac{\hat{\theta}_k^2}{2} \right)^{-(a_0 + \frac{1}{2})} \tag{4.26}$$

is obtained. At first glance, it looks like the problem of specifying γ_k has been replaced by the problem of finding good parameter values for a_0, b_0. Looking at Equation (4.24) more closely, it can be observed that for $a_0, b_0 \to 0$, $\Gamma(\gamma_k | a_0, b_0)$ is equal to the *noninformative Jeffreys prior* [152]

$$p(\gamma_k) \propto \frac{1}{\gamma_k}. \tag{4.27}$$

It is called noninformative because it is scale invariant, $p(\gamma_k') \propto \frac{1}{\gamma_k'}$ for $\gamma_k' = c\gamma_k$ with an arbitrary positive constant c. Moreover, it is flat in log space,

$$p(\log \gamma_k) = \begin{cases} \frac{1}{\log c_u - \log c_l} \\ 0 \end{cases} \quad \text{if} \quad p(\gamma_k) = \begin{cases} \frac{1}{\gamma_k} & \text{for} \quad 0 < c_l \leq \gamma_k \leq c_u, \\ 0 & \text{else.} \end{cases} \tag{4.28}$$

Although this kind of prior looks appealing from a theoretical point of view, it is often unexpedient in practical settings because, apart from the fact that it is an *improper prior*, its log is non-concave and it therefore promotes multimodality in the posterior distribution $p(\hat{\boldsymbol{\theta}} | \mathcal{D})$ [153]. This can be seen assuming a single-parameter model with

Gaussian likelihood of the form

$$\log \mathcal{L}(\hat{\theta}|\mathcal{D}) \propto -\frac{1}{2\sigma^2}(\hat{\theta} - \mu_{ML})^2. \tag{4.29}$$

The log posterior is

$$\log p(\hat{\theta}|\mathcal{D}) \propto -\frac{1}{2\sigma^2}(\hat{\theta} - \mu_{ML})^2 - \left(a_0 + \frac{1}{2}\right) \log \left(b_0 + \frac{\hat{\theta}^2}{2}\right) \tag{4.30}$$

and its first derivative is

$$\frac{\partial}{\partial\hat{\theta}} \log p(\hat{\theta}|\mathcal{D}) = -\frac{1}{\sigma^2}(\hat{\theta} - \mu_{ML}) - \left(a_0 + \frac{1}{2}\right) \left(b_0 + \frac{\hat{\theta}^2}{2}\right)^{-1} \hat{\theta}, \tag{4.31}$$

i.e., there are up to three roots/two potential maxima. The log Student-t prior and the posterior using a likelihood of the form $\mathcal{L}(\mathcal{D}|\hat{\boldsymbol{\theta}}) \propto -\frac{3}{2}\|\hat{\boldsymbol{\theta}} - \left(\begin{smallmatrix}-1\\-1\end{smallmatrix}\right)\|^2$ are plotted for the case of two model parameters $\hat{\boldsymbol{\theta}} = \left(\begin{smallmatrix}\hat{\theta}_0\\\hat{\theta}_1\end{smallmatrix}\right)$ in the fourth column of Figure 4.4. It can be observed that local optima are induced along the coordinate axes.

Figure 4.5 shows the same regression problem as in the previous subsection, but using a Student-t prior distribution with $a_0 = b_0 = 10^{-10}$ instead of a Laplacian prior. The posterior maximization is performed using the EM algorithm (see Section 4.3.1) as described in [147].

The relevance vector machine

Instead of putting another prior on the hyperparameters γ_k and average, another viable strategy is to integrate over the model parameters $\hat{\boldsymbol{\theta}}$ to get the *marginal likelihood* or *evidence*

$$p(\mathcal{D}|\gamma, \hat{\sigma}^2) = \int \mathcal{L}(\mathcal{D}|\hat{\boldsymbol{\theta}}, \hat{\sigma}^2) p(\hat{\boldsymbol{\theta}}|\gamma) d\hat{\boldsymbol{\theta}} \tag{4.32}$$

and find the optimal values for γ and the likelihood variance $\hat{\sigma}^2$ by performing *type-II maximum likelihood* or *evidence approximation* [154, 155]

$$\gamma^*, \hat{\sigma}^{2,*} = \arg\max_{\gamma, \hat{\sigma}^2} \log p(\mathcal{D}|\gamma, \hat{\sigma}^2). \tag{4.33}$$

This is the central principle of the *relevance vector machine* (RVM) [153, 156, 157].

One way to solve Equation (4.33) is to apply the EM algorithm, see Section 4.3.1 for a general introduction. Using Jensen's inequality [158], the log evidence in Equation (4.32) can be lower bounded as

$$\begin{aligned} \log p(\mathcal{D}|\gamma, \hat{\sigma}^2) &= \log \int \mathcal{L}(\mathcal{D}|\hat{\boldsymbol{\theta}}, \hat{\sigma}^2) p(\hat{\boldsymbol{\theta}}|\gamma) d\hat{\boldsymbol{\theta}} \\ &\geq \int q(\hat{\boldsymbol{\theta}}) \log \left(\frac{\mathcal{L}(\mathcal{D}|\hat{\boldsymbol{\theta}}, \hat{\sigma}^2) p(\hat{\boldsymbol{\theta}}|\gamma)}{q(\hat{\boldsymbol{\theta}})}\right) d\hat{\boldsymbol{\theta}} = \mathcal{F}(q; \gamma, \hat{\sigma}^2), \end{aligned} \tag{4.34}$$

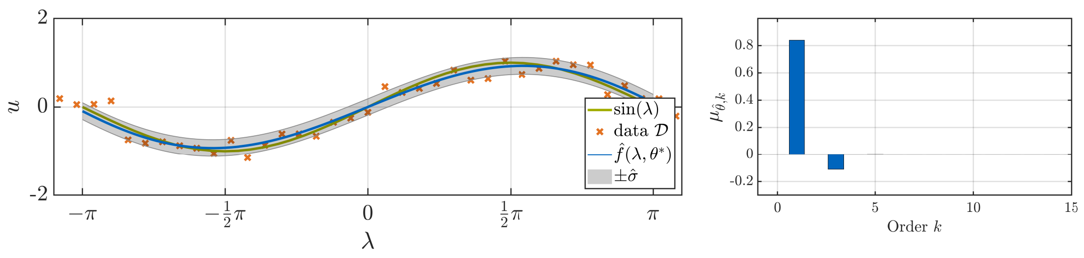


FIGURE 4.6: Regression problem using the relevance vector machine and the same model $\hat{f}(\lambda, \hat{\boldsymbol{\theta}})$ and data as in Figure 4.3. The right part of the figure shows the posterior mode $\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}$ and clearly reveals sparsity.

where $q(\hat{\boldsymbol{\theta}})$ is an arbitrary probability distribution. The strategy of EM is to iteratively maximize the lower bound $\mathcal{F}$ in the auxiliary distribution $q(\hat{\boldsymbol{\theta}})$ and the parameters $\hat{\sigma}^2, \gamma$. The $q(\hat{\boldsymbol{\theta}})$ that maximizes $\mathcal{F}$ given some parameter estimates $\hat{\sigma}^{2,(t)}, \gamma^{(t)}$ is $q^{(t)}(\hat{\boldsymbol{\theta}}) \propto \mathcal{L}(\mathcal{D}|\hat{\boldsymbol{\theta}}, \hat{\sigma}^{2,(t)})p(\hat{\boldsymbol{\theta}}|\gamma^{(t)})$ because the inequality becomes an equality in that case. For linear models as defined by Equation (4.10) and Gaussian noise, $q^{(t)}$ is a Gaussian

$$q^{(t)}(\hat{\boldsymbol{\theta}}) = \mathcal{N}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}^{(t)}, \boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}}^{(t)}) \tag{4.35}$$

where

$$\boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}}^{(t)} = \left(\frac{1}{\hat{\sigma}^{2,(t)}}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \operatorname{diag}(\gamma^{(t)})\right)^{-1} \tag{4.36}$$

$$\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}^{(t)} = \frac{1}{\hat{\sigma}^{2,(t)}}\boldsymbol{\Sigma}^{(t)}\boldsymbol{\Phi}^T\boldsymbol{u} \tag{4.37}$$

with the output data vector $\boldsymbol{u} = \begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix}$. In the next step, the lower bound in Equation (4.34) is maximized in $\hat{\sigma}^2, \gamma$ given $q^{(t)}$ as above. Only keeping terms dependent on γ, the lower bound $\mathcal{F}(q^{(t)}; \gamma, \hat{\sigma}^2)$ is

$$\mathcal{F}(q^{(t)}; \gamma, \hat{\sigma}^2) \propto \langle \log p(\hat{\boldsymbol{\theta}}|\gamma)\rangle_{q^{(t)}} = \frac{1}{2}\sum_{k=0}^{K}\left(\log\gamma_k - \gamma_k\langle\hat{\theta}_k^2\rangle_{q^{(t)}}\right). \tag{4.38}$$

Setting the derivative w.r.t. γ_k to zero yields the update equation

$$\gamma_k^{(t+1)} = \frac{1}{(\mu_{\hat{\boldsymbol{\theta}},k}^{(t)})^2 + \Sigma_{\hat{\boldsymbol{\theta}},kk}^{(t)}}. \tag{4.39}$$

Similarly, we obtain the update equation for $\hat{\sigma}^2$,

$$\hat{\sigma}^{2,(t+1)} = \frac{\|\boldsymbol{u} - \boldsymbol{\Phi}\boldsymbol{\mu}^{(t)}\|^2 + \operatorname{Tr}(\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{(t)}\boldsymbol{\Phi})}{N}. \tag{4.40}$$

The update equations (4.36)–(4.40) are iteratively evaluated until convergence is reached. The posterior $p(\hat{\boldsymbol{\theta}}|\mathcal{D}, \hat{\sigma}^{2,*}, \gamma^*)$ is given by the final value of Equation (4.35)

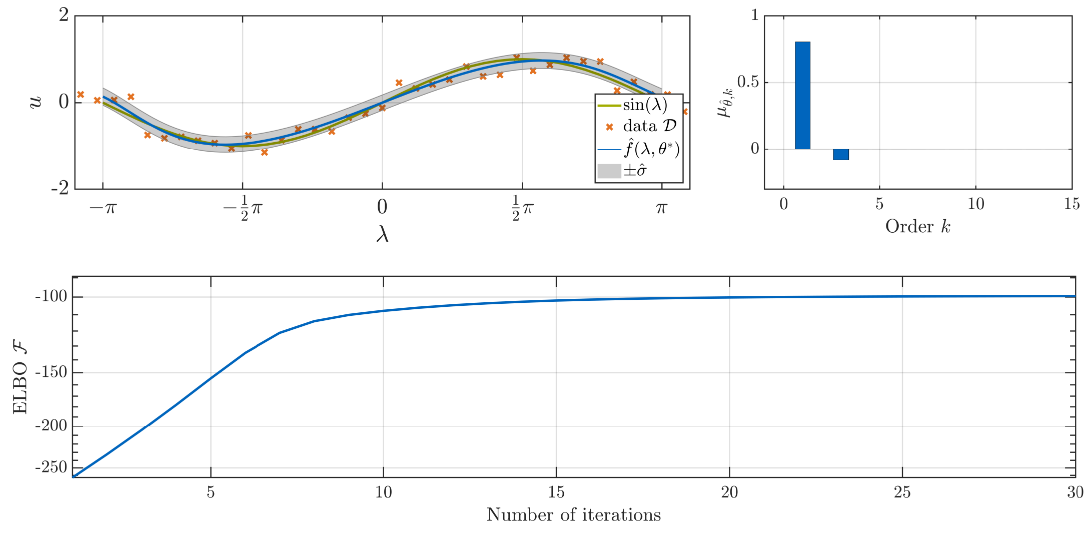


FIGURE 4.7: Regression problem (top) using the variational relevance vector machine (VRVM) and the same model $\hat{f}(\lambda, \hat{\boldsymbol{\theta}})$ and data as in Figure 4.3. The right part of the figure shows the posterior mode $\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}$. Again, the model exhibits sparsity. The bottom plot shows the evidence lower bound (ELBO) $\mathcal{F}$ over training iterations. It can be used to monitor convergence.

and the predictive distribution

$$p(u|\lambda, \mathcal{D}, \boldsymbol{\gamma}^*, \hat{\sigma}^{2,*}) = \mathcal{N}(u|\boldsymbol{\mu}_{\text{pred}}(\lambda), \sigma^2_{\text{pred}}(\lambda)) \tag{4.41}$$

is equivalent to the findings in Equation (4.19)–(4.21).

A regression example together with the posterior mode $\boldsymbol{\mu}^*_{\hat{\boldsymbol{\theta}}}$ is shown in Figure 4.6. It is clearly visible that the posterior mode is sparse, i.e., that most of its components are pruned from the model. The sparsity pattern is proven and thoroughly analyzed in [159, 160].

The variational relevance vector machine

Instead of maximizing the evidence w.r.t. the hyperparameters γ to get the point estimate γ^*, the *variational relevance vector machine* [161, 162] (VRVM) puts a *Gamma* prior on γ and performs a *variational mean-field approximation* (see Section 4.3.4) to find an approximate posterior $q_\gamma(\gamma)$.

To be precise, the prior distributions that are used in the VRVM are

$$p(\hat{\boldsymbol{\theta}}_k|\gamma_k) = \mathcal{N}(\hat{\theta}_k|0, \gamma_k^{-1}), \tag{4.42}$$

$$p(\gamma_k) = \Gamma(\gamma_k|a_0, b_0), \tag{4.43}$$

$$p(\tau) = \Gamma(\tau|c_0, d_0), \tag{4.44}$$

where Γ denotes the *Gamma* distribution as defined in Equation (4.24) and τ is the noise precision defined as $\tau = \hat{\sigma}^{-2}$.

The posterior $p(\hat{\boldsymbol{\theta}}, \gamma, \tau|\mathcal{D})$ is thus

$$p(\hat{\boldsymbol{\theta}}, \gamma, \tau|\mathcal{D}) \propto \mathcal{L}(\mathcal{D}|\hat{\boldsymbol{\theta}}, \tau) \prod_{k=0}^{K} \left(\mathcal{N}(\hat{\theta}_k|0, \gamma_k^{-1}) \Gamma(\gamma_k|a_0, b_0) \right) \Gamma(\tau|c_0, d_0). \tag{4.45}$$

The variational mean field approximation attempts to find an approximate distribution $q(\hat{\boldsymbol{\theta}}, \gamma, \tau)$ to $p(\hat{\boldsymbol{\theta}}, \gamma, \tau|\mathcal{D})$ of the factorial form

$$q(\hat{\boldsymbol{\theta}}, \gamma, \tau) = q_{\hat{\boldsymbol{\theta}}}(\hat{\boldsymbol{\theta}}) q_\gamma(\gamma) q_\tau(\tau). \tag{4.46}$$

This is achieved by minimization of the *Kullback-Leibler* (KL) *divergence* [163, 164]

$$D_{\mathrm{KL}}(q(\hat{\boldsymbol{\theta}}, \gamma, \tau) \; || \; p(\hat{\boldsymbol{\theta}}, \gamma, \tau|\mathcal{D})) = \int q(\hat{\boldsymbol{\theta}}, \gamma, \tau) \log \frac{q(\hat{\boldsymbol{\theta}}, \gamma, \tau)}{p(\hat{\boldsymbol{\theta}}, \gamma, \tau|\mathcal{D})} d\hat{\boldsymbol{\theta}} d\gamma d\tau. \tag{4.47}$$

In Section 4.3.4, it is derived that the distribution $q_{\theta_k}(\theta_k)$ that minimizes this KL-divergence holding all other $q_{\theta_j}(\theta_j), j \neq k$ fixed is

$$q_{\theta_k}(\theta_k) = \frac{\exp\left\{ \langle \log p(\hat{\boldsymbol{\theta}}, \gamma, \tau, \mathcal{D}) \rangle_{i \neq k} \right\}}{\int \exp\left\{ \langle \log p(\hat{\boldsymbol{\theta}}, \gamma, \tau, \mathcal{D}) \rangle_{i \neq k} \right\} d\theta_k} \tag{4.48}$$

where $\langle \cdot \rangle_{i \neq k}$ denotes expectation w.r.t. $\prod_{i \neq k} q_{\theta_k}(\theta_k)$ and θ_k stands for any of the parameters $\hat{\boldsymbol{\theta}}, \gamma_k, \tau$. It is noted that $q_{\theta_k}(\theta_k)$ implicitly depends on all other $q_{\theta_j}(\theta_j)$, so that all q's need to be iteratively updated until convergence.

Due to the choice of conjugate *Gamma* priors for the precision parameters γ, τ, the approximate distributions $q_{\hat{\boldsymbol{\theta}}}(\hat{\boldsymbol{\theta}}), q_\gamma(\gamma), q_\tau(\tau)$ are given in closed form (see [161]),

$$q_{\hat{\boldsymbol{\theta}}} = \mathcal{N}(\hat{\boldsymbol{\theta}}|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}, \boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}}), \qquad q_{\gamma_k}(\gamma_k) = \Gamma(\gamma_k|\tilde{a}, \tilde{b}_k), \qquad q_\tau(\tau) = \Gamma(\tau|\tilde{c}, \tilde{d}), \tag{4.49}$$

with

$$\boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}} = \left(\langle \tau \rangle \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \mathrm{diag}(\langle \gamma \rangle) \right)^{-1}, \qquad \boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}} = \langle \tau \rangle \boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}} \boldsymbol{\Phi}^T \boldsymbol{u}, \tag{4.50}$$

$$\tilde{a} = a_0 + \frac{1}{2}, \qquad \tilde{b}_k = b_0 + \frac{1}{2} \langle \hat{\theta}_k^2 \rangle, \tag{4.51}$$

$$\tilde{c} = c_0 + \frac{N}{2}, \qquad \tilde{d} = d_0 + \frac{1}{2} \langle (\boldsymbol{u} - \boldsymbol{\Phi}\hat{\boldsymbol{\theta}})^T (\boldsymbol{u} - \boldsymbol{\Phi}\hat{\boldsymbol{\theta}}) \rangle. \tag{4.52}$$

The required expected values (all w.r.t. $\langle \cdot \rangle_{i \neq k}$) are

$$\langle \hat{\boldsymbol{\theta}} \rangle = \boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}, \qquad \langle \hat{\boldsymbol{\theta}} \hat{\boldsymbol{\theta}}^T \rangle = \boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}} + \boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}} \boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}^T, \tag{4.53}$$

$$\langle \gamma_k \rangle = \frac{\tilde{a}}{\tilde{b}_k}, \qquad \langle \tau \rangle = \frac{\tilde{c}}{\tilde{d}}. \tag{4.54}$$

After convergence, the posterior can be approximated with

$$p(\hat{\boldsymbol{\theta}}, \tau|\mathcal{D}) \approx q_{\hat{\boldsymbol{\theta}}}(\hat{\boldsymbol{\theta}}) q_\tau(\tau) \tag{4.55}$$

so that the predictive distribution becomes

$$p(u|\lambda, \mathcal{D}) = \int \mathcal{N}(u|\boldsymbol{\varphi}^T(\lambda)\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}, \tau^{-1}) q_{\hat{\boldsymbol{\theta}}}(\hat{\boldsymbol{\theta}}) q_\tau(\tau) d\hat{\boldsymbol{\theta}} d\tau. \tag{4.56}$$

Simultaneous integration over $\hat{\boldsymbol{\theta}}$ and τ is analytically intractable. However, the variance $\langle \tau^2 \rangle - \langle \tau \rangle^2 = \frac{\tilde{c}}{\tilde{d}^2} \sim \mathcal{O}(N^{-1})$ for large N [161], so that it is valid to approximate $q_\tau(\tau) \approx \delta(\tau - \langle \tau \rangle)$ for sufficiently large N. Thus, the predictive distribution can be approximated as

$$\begin{aligned} p(u|\lambda, \mathcal{D}) &= \int \mathcal{N}(u|\boldsymbol{\varphi}^T(\lambda)\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}, \langle \tau \rangle^{-1}) q_{\hat{\boldsymbol{\theta}}}(\hat{\boldsymbol{\theta}}) d\hat{\boldsymbol{\theta}} \\ &= \mathcal{N}(u|\mu_{\text{pred}}(\lambda), \sigma^2_{\text{pred}}(\lambda)), \end{aligned} \tag{4.57}$$

with

$$\mu_{\text{pred}} = \boldsymbol{\varphi}^T(\lambda)\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}, \tag{4.58}$$

$$\sigma^2_{\text{pred}}(\lambda) = \langle \tau \rangle^{-1} + \boldsymbol{\varphi}^T(\lambda)\boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}}\boldsymbol{\varphi}(\lambda). \tag{4.59}$$

Figure 4.7 shows a regression example using the same data and model $\hat{f}(\lambda, \hat{\boldsymbol{\theta}})$ as before. Again, the model parameters $\hat{\boldsymbol{\theta}}$ exhibit sparsity. Moreover, the VRVM allows to compute the *evidence lower bound* (ELBO) which is an approximation to the model evidence and allows to monitor convergence of the training process as well as to compare how well different kinds of models (e.g., with a different number of basis functions $\varphi_j(\lambda)$) can represent the data $\mathcal{D}$. The basic principle behind that are explained in the next subsection.

4.2.3 Bayesian model comparison

In the context of Bayes' law in Section 4.2.2, the *model evidence* $p(\mathcal{D})$ is introduced as the normalization constant to the posterior

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{\mathcal{L}(\mathcal{D}|\boldsymbol{\theta}) p_0(\boldsymbol{\theta})}{p(\mathcal{D})}. \tag{4.60}$$

This normalization constant is found by integrating the nominator $\mathcal{L}(\mathcal{D}|\boldsymbol{\theta}) p_0(\boldsymbol{\theta})$ over the parameters

$$p(\mathcal{D}) = \int \mathcal{L}(\mathcal{D}|\boldsymbol{\theta}) p_0(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{4.61}$$

Apart from being a normalization constant, the model evidence $p(\mathcal{D})$ can be viewed as the *expected likelihood* under the prior

$$p(\mathcal{D}) = \int \mathcal{L}(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} = \langle \mathcal{L}(\mathcal{D}|\boldsymbol{\theta}) \rangle_{p_0(\boldsymbol{\theta})}. \tag{4.62}$$

Seen from this perspective, $p(\mathcal{D})$ can be interpreted as being a measure of how well the model (encoded in the likelihood function $\mathcal{L}(\mathcal{D}|\boldsymbol{\theta})$) fits the data $\mathcal{D}$ if parameters

$\boldsymbol{\theta}$ were sampled randomly from the prior $p_0(\boldsymbol{\theta})$. However, more often than not, the model evidence $p(\mathcal{D})$ is not computable in closed form.

In variational approximations (see Section 4.2.2, Section 4.3.4), the objective is to find a distribution $q^*(\boldsymbol{\theta})$ out of a family of distributions $q(\boldsymbol{\theta})$ that best approximates the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ in the sense of minimum KL divergence,

$$\begin{aligned} q^*(\boldsymbol{\theta}) &= \arg\min_q D_{\mathrm{KL}}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})) \\ &= \arg\min_q \int q(\boldsymbol{\theta}) \log\left(\frac{p(\boldsymbol{\theta}|\mathcal{D})}{q(\boldsymbol{\theta})}\right) d\boldsymbol{\theta}. \end{aligned} \tag{4.63}$$

One can make the observation that

$$\begin{aligned} \log p(\mathcal{D}) &= \int q(\boldsymbol{\theta}) \log p(\mathcal{D}) d\boldsymbol{\theta} \\ &= \int q(\boldsymbol{\theta}) \left[\log(p(\boldsymbol{\theta}|\mathcal{D})p(\mathcal{D})) - \log p(\boldsymbol{\theta}|\mathcal{D}) + \log q(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta})\right] d\boldsymbol{\theta} \\ &= \int q(\boldsymbol{\theta}) \left[\log\left(\frac{p(\boldsymbol{\theta},\mathcal{D})}{q(\boldsymbol{\theta})}\right) + \log\left(\frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})}\right)\right] d\boldsymbol{\theta} \\ &= \mathcal{F}[q(\boldsymbol{\theta}), p(\boldsymbol{\theta},\mathcal{D})] + D_{\mathrm{KL}}(q||p(\boldsymbol{\theta}|\mathcal{D})) \end{aligned} \tag{4.64}$$

with the *evidence lower bound* (ELBO)

$$\mathcal{F}[q(\boldsymbol{\theta}), p(\boldsymbol{\theta},\mathcal{D})] = \int q(\boldsymbol{\theta}) \log\left(\frac{p(\boldsymbol{\theta},\mathcal{D})}{q(\boldsymbol{\theta})}\right) d\boldsymbol{\theta}. \tag{4.65}$$

For the KL divergence, it is known that $D_{\mathrm{KL}}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})) \geq 0$ so that the ELBO $\mathcal{F}$ represents a rigorous lower bound to the evidence $p(\mathcal{D})$. As in variational approximations, the target is to minimize $D_{\mathrm{KL}}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D}))$ (which is equivalent to maximization of the ELBO $\mathcal{F}$), after convergence, one can make the approximation $D_{\mathrm{KL}}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})) \approx 0$ or

$$\log p(\mathcal{D}) \approx \mathcal{F}[q(\boldsymbol{\theta}), p(\boldsymbol{\theta},\mathcal{D})]. \tag{4.66}$$

The ELBO $\mathcal{F}$ can thus not only be used to monitor training convergence, but also to directly compare different kinds of models, e.g., with different numbers of basis functions $\dim(\boldsymbol{\varphi}(\lambda))$.

4.3 Approximate inference

Within the literature, there is no consistent definition of the notion of *statistical inference* [165–167]. From the writer's point of view, statistical inference is the unification of all methods that enable to draw conclusions on the underlying mechanisms that gave rise to some observed data [168]. As such, it can be segmented into the stages of *parameter estimation/model training*, *model selection* and *prediction* [169].

In all three of the above stages of statistical inference, it is common that expected values, i.e., integrals of the form

$$\langle f \rangle = \int f(\boldsymbol{\lambda}) p(\boldsymbol{\lambda}) d\boldsymbol{\lambda} \tag{4.67}$$

where $f(\boldsymbol{\lambda})$ is an arbitrary function of the random variable $\boldsymbol{\lambda}$, do not have closed form analytical solutions, meaning that *exact inference* is infeasible.

A multitude of approximate integration schemes have been developed over the last decades to solve Equation (4.67), all varying in accuracy and computational complexity. This section is devoted to give an overview over some important *approximate inference* techniques, most of which have found a use in the model presented in Chapter 5 of this work.

4.3.1 Point-based estimates

Point-based estimates to Equation (4.67) are described in many statistics and machine learning text books (e.g., [155, 169, 170]), most commonly in the context of *maximum likelihood* or *maximum a posteriori estimation*. The approximation consists of replacing the distribution $p(\boldsymbol{\lambda})$ by a δ-distribution centered at its mode, i.e.

$$p(\boldsymbol{\lambda}) \approx \delta(\boldsymbol{\lambda} - \boldsymbol{\lambda}^*), \tag{4.68}$$

where

$$\boldsymbol{\lambda}^* = \arg\max_{\lambda} p(\boldsymbol{\lambda}). \tag{4.69}$$

The approximation to Equation (4.67) then becomes

$$\langle f(\boldsymbol{\lambda}) \rangle \approx \int f(\boldsymbol{\lambda}) \delta(\boldsymbol{\lambda} - \boldsymbol{\lambda}^*) d\boldsymbol{\lambda} = f(\boldsymbol{\lambda}^*). \tag{4.70}$$

If, after observation of some data $\mathcal{D}$, $p(\boldsymbol{\lambda})$ plays the role of a likelihood function, i.e., if $p(\boldsymbol{\lambda}) \propto \mathcal{L}(\mathcal{D}|\boldsymbol{\lambda})$, Equation (4.69) is known as *maximum likelihood* (ML) estimation. If $p(\boldsymbol{\lambda}|\mathcal{D}) \propto \mathcal{L}(\mathcal{D}|\boldsymbol{\lambda}) p_0(\boldsymbol{\lambda})$ is a posterior distribution (with an arbitrary prior p_0), the framework is called *maximum a posteriori* (MAP) estimation.

Point estimates tend to be inaccurate particularly when $p(\boldsymbol{\lambda})$ is spread out, and $p(\boldsymbol{\lambda})$ and/or $f(\boldsymbol{\lambda})$ are of high complexity. A major drawback is that no error estimates for Equation (4.70) are given.

The optimization problem in Equation (4.69) can be solved by a suitable optimization algorithm [171–173]. A broadly applicable and robust method to find MAP estimates in latent variable models is given by the *expectation-maximization* (EM) algorithm [174, 175] which is introduced in the next subsection.

Expectation-maximization algorithm

The expectation-maximization algorithm [174, 175] is an efficient way to find ML/ MAP estimates of latent variable models by iteratively estimating expected values under an auxiliary distribution $q(\boldsymbol{z})$ over the latent variables $\boldsymbol{z}$ and subsequently performing a maximization step given the current expected values under $q(\boldsymbol{z})$.

Assume the joint distribution $p(\boldsymbol{\lambda}, \boldsymbol{z})$ over some observed and latent variables $\boldsymbol{\lambda}$ and $\boldsymbol{z}$, respectively. The marginal distribution over the observed variables is

$$p(\boldsymbol{\lambda}) = \int p(\boldsymbol{\lambda}, \boldsymbol{z}) d\boldsymbol{z} \tag{4.71}$$

and the objective is to find the mode $\boldsymbol{\lambda}^*$ of $p(\boldsymbol{\lambda})$, i.e.,

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} p(\boldsymbol{\lambda}) = \arg\max_{\boldsymbol{\lambda}} \int p(\boldsymbol{\lambda}, \boldsymbol{z}) d\boldsymbol{z}. \tag{4.72}$$

Such problems typically arise in MAP estimations of latent variable models, i.e., problems of the form

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} p(\boldsymbol{\lambda}|\mathcal{D}) = \arg\max_{\boldsymbol{\lambda}} \int \mathcal{L}(\mathcal{D}|\boldsymbol{\lambda}, \boldsymbol{z}) p_0(\boldsymbol{\lambda}, \boldsymbol{z}) d\boldsymbol{z} \tag{4.73}$$

where $\mathcal{L}(\mathcal{D}|\boldsymbol{\lambda}, \boldsymbol{z})$ is the likelihood function and $p_0(\boldsymbol{\lambda}, \boldsymbol{z})$ is a prior on the observed and latent parameters $\boldsymbol{\lambda}, \boldsymbol{z}$.

Using Jensen's inequality [158], the (log-)marginal $\log p(\boldsymbol{\lambda})$ can be bounded from below as

$$\begin{aligned}
\log p(\boldsymbol{\lambda}) &= \log \int p(\boldsymbol{\lambda}, \boldsymbol{z}) d\boldsymbol{z} \\
&\geq \int q(\boldsymbol{z}) \log \left(\frac{p(\boldsymbol{\lambda}, \boldsymbol{z})}{q(\boldsymbol{z})} \right) d\boldsymbol{z} \\
&= \langle \log p(\boldsymbol{\lambda}, \boldsymbol{z}) \rangle_q - \langle \log q(\boldsymbol{z}) \rangle_q = \mathcal{F}(q; \boldsymbol{\lambda})
\end{aligned} \tag{4.74}$$

where $\mathcal{F}(q; \boldsymbol{\lambda})$ denotes the lower bound functional and $\langle \cdot \rangle_q$ means expectation w.r.t. $q(\boldsymbol{z})$. The basic strategy of the EM algorithm is to repeat the following two steps until convergence:

E-step: Given the current best estimate $\boldsymbol{\lambda}^{(t)}$ to the MAP value $\boldsymbol{\lambda}^*$, find the optimal distribution $q^{(t)}(\boldsymbol{z})$ that maximizes the lower bound $\mathcal{F}(q; \boldsymbol{\lambda}^{(t)})$. Compute the expected value $\langle p(\boldsymbol{\lambda}, \boldsymbol{z}) \rangle_{q^{(t)}}$.

M-step: Given the current estimate $q^{(t)}(\boldsymbol{z})$ for the distribution q (and the corresponding expected values), find the parameters $\boldsymbol{\lambda}^{(t+1)}$ that maximize the lower bound $\mathcal{F}(q^{(t)}; \boldsymbol{\lambda})$.

Algorithm 1 : The expectation-maximization algorithm

```
Input : $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}^{(0)}$ ;                    // Initialization
1  $t \leftarrow 0$
2  while (not converged) do
3      E-step: Compute
4              $q^{(t)}(\boldsymbol{z}) \propto p(\boldsymbol{\lambda}^{(t)}, \boldsymbol{z})$;
5              and estimate
6              $\langle \log p(\boldsymbol{\lambda}, \boldsymbol{z}) \rangle_{q^{(t)}}$;
7              with, e.g., Monte Carlo
8      M-step: Maximize
9              $\boldsymbol{\lambda}^{(t+1)} = \arg\max_{\boldsymbol{\lambda}} \langle \log p(\boldsymbol{\lambda}, \boldsymbol{z}) \rangle_{q^{(t)}}$;
10             with a suitable stochastic optimization algorithm
11     $t \leftarrow t + 1$;
12 end
13 return $\boldsymbol{\lambda}^*$ ;                    // Optimizer to Equation (4.72)
```

It can readily be observed that the $q^{(t)}(\boldsymbol{z})$ maximizing the lower bound $\mathcal{F}(q; \boldsymbol{\lambda}^{(t)})$ is given by

$$q^{(t)}(\boldsymbol{z}) \propto p(\boldsymbol{\lambda}^{(t)}, \boldsymbol{z}), \tag{4.75}$$

as the lower bound becomes tight, i.e., the inequality in Equation (4.74) becomes an equality in that case. Expected values w.r.t. q are generally not given in closed form and need to be estimated with, e.g., Monte Carlo (see Section 4.3.3). Also, the (sub)optimal distribution $q^{(t)}(\boldsymbol{z})$ may be searched within a family of distributions $q(\boldsymbol{z}|\boldsymbol{\xi})$ parametrized by a finite number of parameters $\boldsymbol{\xi}$, leading to a potentially non-zero lower bound $\mathcal{F}$ after the E-step. This strategy is closely related to variational inference which will be discussed in Section 4.3.4.

Given an estimate $q^{(t)}(\boldsymbol{z})$, the M-step consists of the optimization problem

$$\boldsymbol{\lambda}^{(t+1)} = \arg\max_{\boldsymbol{\lambda}} \mathcal{F}(q^{(t)}; \boldsymbol{\lambda}) = \arg\max_{\boldsymbol{\lambda}} \langle \log p(\boldsymbol{\lambda}, \boldsymbol{z}) \rangle_{q^{(t)}} \tag{4.76}$$

which can be solved using standard stochastic optimization algorithms [171–173], see also Section 4.4. Both E- and M-step are run alternately until convergence of $\boldsymbol{\lambda}$. It is noted that neither the E- nor the M-step need to be driven to convergence in every single iteration t. Rather, it is sufficient to improve in every iteration in the sense that $\log p(\boldsymbol{\lambda}^{(t+1)}) \geq \log p(\boldsymbol{\lambda}^{(t)})$. The EM-algorithm is summarized in Algorithm 1.

4.3.2 Laplace approximation

The Laplace approximation [176] $p_{LA}(\boldsymbol{\lambda})$ to $p(\boldsymbol{\lambda})$ is given by a Gaussian centered around a mode of $p(\boldsymbol{\lambda})$:

$$p_{LA}(\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{\lambda}|\boldsymbol{\mu}_{LA}, \boldsymbol{\Sigma}_{LA}), \tag{4.77}$$

where the mean $\boldsymbol{\mu}_{LA}$ is found by

$$\nabla_{\boldsymbol{\lambda}} p(\boldsymbol{\lambda})|_{\boldsymbol{\lambda}=\boldsymbol{\mu}_{LA}} = \mathbf{0} \tag{4.78}$$

and the covariance $\boldsymbol{\Sigma}_{LA}$ is

$$\boldsymbol{\Sigma}_{LA} = -\nabla_{\boldsymbol{\lambda}} \nabla_{\boldsymbol{\lambda}} \log p(\boldsymbol{\lambda}), \tag{4.79}$$

such that $p_{LA}(\boldsymbol{\lambda})$ has the same curvature as $p(\boldsymbol{\lambda})$ at the mode $\boldsymbol{\lambda} = \boldsymbol{\mu}_{LA}$. Next, the integral in Equation (4.67) is approximated as

$$\langle \boldsymbol{f}(\boldsymbol{\lambda}) \rangle \approx \int \boldsymbol{f}(\boldsymbol{\lambda}) p_{LA}(\boldsymbol{\lambda}) d\boldsymbol{\lambda} \tag{4.80}$$

which is often analytically solvable since the moments of a Gaussian are given in closed form [177]. For non-analytical quantities of interest $\boldsymbol{f}(\boldsymbol{\lambda})$, Equation (4.80) can efficiently be approximated by direct sampling from the Gaussian $p_{LA}(\boldsymbol{\lambda})$.

A disadvantage of the Laplace approximation is that it only makes use of local information at a mode of the original distribution $p(\boldsymbol{\lambda})$ and is therefore not capable to capture global characteristics. Furthermore, if $\boldsymbol{\lambda}$ is very high dimensional, computation and storage of the Hessian can be prohibitive, in which case it could be assumed to be diagonal. As for point estimates, approximation errors are hard to quantify.

4.3.3 The Monte Carlo method

Thanks to its versatility, ease of use, and robustness particularly in high stochastic dimensions, the Monte Carlo method [178, 179] is one of the principal workhorses in computational physics and engineering. Moreover, it provides rigorous error estimates and is *unbiased* in the sense that given unlimited computational resources, integrals can be approximated at arbitrary precision. On the other side, Monte Carlo simulations are generally inhibited by their extensive computational costs. The present section gives a brief introduction to the most important Monte Carlo techniques for UQ.

Simple sampling and statistical error

Using simple sampling Monte Carlo, the integral in Equation (4.67) can be estimated as[6]

$$\begin{aligned} \langle f(\boldsymbol{\lambda}) \rangle &= \int f(\boldsymbol{\lambda}) p(\boldsymbol{\lambda}) d\boldsymbol{\lambda} \\ &\approx \frac{1}{N} \sum_{n=1}^{N} f(\boldsymbol{\lambda}^{(n)}) = \bar{f}, \qquad \boldsymbol{\lambda}^{(n)} \sim p(\boldsymbol{\lambda}). \end{aligned} \tag{4.81}$$

[6]In this subsection, it is assumed that f is a scalar quantity for ease of notation.

The mean squared approximation error $(\Delta f)^2$ can be computed according to

$$(\Delta f)^2 = \left\langle \left(\bar{f} - \langle f(\boldsymbol{\lambda})\rangle\right)^2 \right\rangle = \langle \bar{f}^2 \rangle - \langle f \rangle^2, \tag{4.82}$$

where

$$\begin{aligned} \langle \bar{f}^2 \rangle &= \left\langle \frac{1}{N^2} \sum_{m=1}^{N} \sum_{n=1}^{N} f(\boldsymbol{\lambda}^{(m)}) f(\boldsymbol{\lambda}^{(n)}) \right\rangle \\ &= \left\langle \frac{1}{N^2} \sum_{n=1}^{N} f^2(\boldsymbol{\lambda}^{(n)}) + \frac{1}{N^2} \sum_{m=1}^{N} \sum_{n \neq m} f(\boldsymbol{\lambda}^{(m)}) f(\boldsymbol{\lambda}^{(n)}) \right\rangle \\ &= \frac{1}{N} \langle f^2 \rangle + \frac{N-1}{N} \langle f(\boldsymbol{\lambda}) \rangle^2 . \end{aligned} \tag{4.83}$$

Plugging the result in to Equation (4.82) gives

$$(\Delta f)^2 = \frac{1}{N} \langle f^2 \rangle - \frac{1}{N} \langle f \rangle^2 = \frac{\sigma^2}{N}, \tag{4.84}$$

where σ^2 denotes the variance of $f(\boldsymbol{\lambda})$ under $p(\boldsymbol{\lambda})$, which is approximated by the unbiased sample variance

$$\sigma^2 \approx s^2 = \frac{1}{N-1} \sum_{n=1}^{N} (f(\boldsymbol{\lambda}^{(n)}) - \bar{f})^2. \tag{4.85}$$

This leads to the well-known Monte Carlo error for uncorrelated samples

$$\Delta f = \frac{\sigma}{\sqrt{N}}. \tag{4.86}$$

Importance sampling

Importance sampling allows to estimate the integral given in Equation (4.67) using samples from a different distribution $q(\boldsymbol{\lambda}) \neq p(\boldsymbol{\lambda})$. There are essentially three use cases for importance sampling:

a) It is difficult or impossible to draw samples from $p(\boldsymbol{\lambda})$ directly, but samples from $q(\boldsymbol{\lambda})$ are readily available;

b) The sample variance s^2 can be reduced, i.e., the same Monte Carlo error can be achieved with potentially much less samples and therefore less computational effort;

c) Only samples of $f(\boldsymbol{\lambda})$ under the distribution $q(\boldsymbol{\lambda})$ are given – samples can be 'reweighed' to estimate statistics under $p(\boldsymbol{\lambda})$.

The key idea of importance sampling is to rewrite Equation (4.67) as

$$\langle f(\boldsymbol{\lambda}) \rangle = \int f(\boldsymbol{\lambda}) p(\boldsymbol{\lambda}) d\boldsymbol{\lambda} = \int f(\boldsymbol{\lambda}) \frac{p(\boldsymbol{\lambda})}{q(\boldsymbol{\lambda})} q(\boldsymbol{\lambda}) d\boldsymbol{\lambda}, \tag{4.87}$$

where $q(\boldsymbol{\lambda})$ is a valid probability distribution easy to sample from. The importance sampling estimate for $\langle f \rangle$ is

$$\langle f(\boldsymbol{\lambda}) \rangle \approx \frac{1}{N} \sum_{n=1}^{N} f(\boldsymbol{\lambda}^{(n)}) \frac{p(\boldsymbol{\lambda}^{(n)})}{q(\boldsymbol{\lambda}^{(n)})}, \qquad \boldsymbol{\lambda}^{(n)} \sim q(\boldsymbol{\lambda}). \tag{4.88}$$

Clearly, no samples of $p(\boldsymbol{\lambda})$ are needed (s. point a)). If it is possible to find and use a distribution $q(\boldsymbol{\lambda})$ s.t. the quantity $f(\boldsymbol{\lambda})\frac{p(\boldsymbol{\lambda})}{q(\boldsymbol{\lambda})}$ shows only small variation, i.e., small sample variance under $q(\boldsymbol{\lambda})$, the Monte Carlo error given in Equation (4.86) can be reduced considerably compared to simple sampling (s. point b)). If only a set of samples

$$\mathcal{D} = \{\boldsymbol{\lambda}^{(n)}, f(\boldsymbol{\lambda}^{(n)})\}_{n=1}^{N}, \qquad \boldsymbol{\lambda}^{(n)} \sim q(\boldsymbol{\lambda}), \tag{4.89}$$

under distribution q are given, samples can be reweighed by the factor $\frac{p(\boldsymbol{\lambda})}{q(\boldsymbol{\lambda})}$ and the expected value $\langle f(\boldsymbol{\lambda}) \rangle$ under p can be computed using Equation (4.88) (point c)).

Markov chain Monte Carlo

In importance sampling, it is of paramount interest to find a distribution $q(\boldsymbol{\lambda})$ that leads to low variance of $f(\boldsymbol{\lambda})\frac{p(\boldsymbol{\lambda})}{q(\boldsymbol{\lambda})}$ under q, i.e., $q(\boldsymbol{\lambda})$ should be as 'similar' as possible to $f(\boldsymbol{\lambda})p(\boldsymbol{\lambda})$. Particularly in high stochastic dimensions $\dim(\boldsymbol{\lambda}) \gg 1$, it is difficult to find such distributions which at the same time need to be straightforward to sample from. In the common case where $p(\boldsymbol{\lambda})$ cannot be sampled from in a non-iterative way (e.g., thermal distributions of complex physical systems or the posterior on the parameters of a neural network), even simple sampling (s. Section 4.3.3) is infeasible.

Markov chain Monte Carlo (MCMC) [180–182] fills this gap in providing a universally usable method to draw samples from any probability distribution p (given ergodicity of the constructed Markov chain). MCMC can be seen as a simulation of a stochastic process that produces a sequence of samples $\boldsymbol{\lambda}^{(1)}, \ldots, \boldsymbol{\lambda}^{(t)}, \boldsymbol{\lambda}^{(t+1)}, \ldots$ at discrete times t generated by a transition kernel w obeying the Markov property

$$w(\boldsymbol{\lambda}^{(t+1)} | \boldsymbol{\lambda}^{(t)}, \ldots, \boldsymbol{\lambda}^{(1)}) = w(\boldsymbol{\lambda}^{(t+1)} | \boldsymbol{\lambda}^{(t)}) \tag{4.90}$$

that defines the transition probability to go from sample/state $\boldsymbol{\lambda}^{(t)}$ to $\boldsymbol{\lambda}^{(t+1)}$. This transition process is designed in such a way that the samples $\boldsymbol{\lambda}^{(t)}$ are asymptotically distributed according to the stationary distribution $p(\boldsymbol{\lambda})$ (but **not** i.i.d.). The most popular class of MCMC samplers is based on the *Metropolis-Hastings* (MH) *algorithm* [183, 184], which is explained in the sequel.

Metropolis-Hastings algorithm The derivation of the Metropolis-Hastings algorithm starts with the *reversibility* or *detailed balance* assumption

$$p(\boldsymbol{\lambda}) w(\boldsymbol{\lambda}' | \boldsymbol{\lambda}) = p(\boldsymbol{\lambda}') w(\boldsymbol{\lambda} | \boldsymbol{\lambda}') \tag{4.91}$$

Algorithm 2 : The Metropolis-Hastings algorithm

```
Input : $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}^{(0)}$ ;                    // Initialization
for $t \leftarrow 0$ to $N$ do
    Sample $\boldsymbol{\lambda} \sim Prop(\boldsymbol{\lambda}|\boldsymbol{\lambda}^{(t)})$;
    Sample $r \sim U(0,1)$;

    if $r < Acc(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(t)})$ ;                    // Equation (4.94)
     then
        $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}$;
    else
        $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)}$;
    end
end
return $\boldsymbol{\lambda}^{(0)}, \ldots, \boldsymbol{\lambda}^{(t)}, \ldots, \boldsymbol{\lambda}^{(N)}$ ;      // Sequence of samples from $p(\boldsymbol{\lambda})$
```

which is a sufficient (but not necessary) condition for the Markov process to have a unique stationary distribution, here denoted by $p(\boldsymbol{\lambda})$. To construct a transition kernel w that complies with Equation (4.91) and that allows to easily simulate the Markov chain, w is split into a *proposition* and *acceptance* step,

$$w(\boldsymbol{\lambda}'|\boldsymbol{\lambda}) = Prop(\boldsymbol{\lambda}'|\boldsymbol{\lambda})Acc(\boldsymbol{\lambda}', \boldsymbol{\lambda}), \tag{4.92}$$

i.e., being in state $\boldsymbol{\lambda}$, a new state $\boldsymbol{\lambda}'$ is proposed (i.e., sampled) according to the predefined probability distribution $Prop(\boldsymbol{\lambda}'|\boldsymbol{\lambda})$ and then accepted with probability $Acc(\boldsymbol{\lambda}', \boldsymbol{\lambda})$ to ensure Equation (4.91). Plugging that in yields

$$p(\boldsymbol{\lambda})Prop(\boldsymbol{\lambda}'|\boldsymbol{\lambda})Acc(\boldsymbol{\lambda}', \boldsymbol{\lambda}) = p(\boldsymbol{\lambda}')Prop(\boldsymbol{\lambda}|\boldsymbol{\lambda}')Acc(\boldsymbol{\lambda}, \boldsymbol{\lambda}'). \tag{4.93}$$

Using the $\boldsymbol{\lambda} \leftrightarrow \boldsymbol{\lambda}'$ symmetry, it is easy to verify that Equation (4.93) is fulfilled for

$$Acc(\boldsymbol{\lambda}', \boldsymbol{\lambda}) = \min\left(1, \frac{p(\boldsymbol{\lambda}')}{p(\boldsymbol{\lambda})} \frac{Prop(\boldsymbol{\lambda}|\boldsymbol{\lambda}')}{Prop(\boldsymbol{\lambda}'|\boldsymbol{\lambda})}\right). \tag{4.94}$$

The Metropolis-Hastings algorithm is summarized in Algorithm 2. Note that only relative densities $\frac{p(\boldsymbol{\lambda}')}{p(\boldsymbol{\lambda})}$ are required, i.e., there is no need to compute the normalization constant of the stationary distribution $p(\boldsymbol{\lambda})$. It is obvious that according to Algorithm 2, the sequence of samples $\boldsymbol{\lambda}^{(0)}, \ldots, \boldsymbol{\lambda}^{(t)}, \ldots, \boldsymbol{\lambda}^{(N)}$ is correlated, particularly when $Prop(\boldsymbol{\lambda}|\boldsymbol{\lambda}^{(t)})$ is centered around $\boldsymbol{\lambda}^{(t)}$.

Due to this correlation, the *effective sample size* $\mathrm{N_{eff}}$ [181, 185] is smaller than the number of Markov chain iterations N. To see this, we recompute the Monte Carlo error Δf as in Section 4.3.3, this time accounting for non-zero correlations between $f(\boldsymbol{\lambda}^{(t)})$ and $f(\boldsymbol{\lambda}^{(t+\Delta t)})$. Combining Equation (4.82) and Equation (4.83) yields

$$
\begin{aligned}
(\Delta f)^2 &= \frac{1}{N^2}\sum_{m=1}^{N}\sum_{n=1}^{N}\left(\langle f(\boldsymbol{\lambda}^{(m)})f(\boldsymbol{\lambda}^{(n)})\rangle - \langle f\rangle^2\right) \\
&= \frac{1}{N^2}\sum_{n=1}^{N}\langle f^2(\boldsymbol{\lambda}^{(n)})\rangle - \frac{1}{N}\langle f\rangle^2 + \frac{2}{N^2}\sum_{n=1}^{N}\sum_{m=n+1}^{N}\left(\langle f(\boldsymbol{\lambda}^{(m)})f(\boldsymbol{\lambda}^{(n)})\rangle - \langle f\rangle^2\right) \\
&= \frac{\sigma^2}{N} + \frac{2}{N^2}\sum_{t=1}^{N}\sum_{\Delta t=1}^{N-n}\left(\langle f(\boldsymbol{\lambda}^{(t)})f(\boldsymbol{\lambda}^{(t+\Delta t)})\rangle - \langle f\rangle^2\right) \\
&= \frac{\sigma^2}{N} + \frac{2}{N}\sum_{\Delta t=1}^{N-1}\left(\langle f(\boldsymbol{\lambda}^{(t)})f(\boldsymbol{\lambda}^{(t+\Delta t)})\rangle - \langle f\rangle^2\right) \\
&\approx \frac{\sigma^2}{N} + \frac{2}{N}\sum_{\Delta t=1}^{\infty}\left(\langle f(\boldsymbol{\lambda}^{(t)})f(\boldsymbol{\lambda}^{(t+\Delta t)})\rangle - \langle f\rangle^2\right) \\
&= \frac{\sigma^2}{N}\tau_{\text{int}},
\end{aligned}
\tag{4.95}
$$

where

$$
\tau_{\text{int}} = 1 + 2\sum_{\Delta t=1}^{\infty}\frac{\langle f(\boldsymbol{\lambda}^{(t)})f(\boldsymbol{\lambda}^{(t+\Delta t)})\rangle - \langle f\rangle^2}{\sigma^2} \tag{4.96}
$$

is the integrated autocorrelation time [185, 186]. I.e., estimating the quantity f with N correlated MCMC samples leads to the same MC error as using

$$
N_{\text{eff}} = \frac{N}{\tau_{\text{int}}} < N \tag{4.97}
$$

uncorrelated samples. It is therefore of primary interest to construct the transition kernel $w(\boldsymbol{\lambda}^{(t+1)}|\boldsymbol{\lambda}^{(t)})$ in such a way that consecutive samples $\boldsymbol{\lambda}^{(t)}, \boldsymbol{\lambda}^{(t+1)}$ (and therefore also f) are as weakly correlated as possible. This is achieved by keeping the acceptance ratio $Acc(\boldsymbol{\lambda}^{(t+1)}, \boldsymbol{\lambda}^{(t)})$ high while doing as large as possible steps $\Delta\boldsymbol{\lambda} = \langle\boldsymbol{\lambda}^{(t+1)} - \boldsymbol{\lambda}^{(t)}\rangle$. Expected values can be computed according to Equation (4.81), but only retaining every $\tau_{\text{int}}^{\text{th}}$ sample of the Markov chain. Also, the Markov chain needs to thermalize, i.e., convergence from the initial configuration $\boldsymbol{\lambda}^{(0)}$ to a state $\boldsymbol{\lambda}^{(t)}$ representative for $p(\boldsymbol{\lambda})$ needs to be waited for.

Metropolis-adjusted Langevin algorithm One way to keep acceptance ratios high and simultaneously do large steps $\Delta\boldsymbol{\lambda}$ is to make use of gradient information of the stationary distribution $p(\boldsymbol{\lambda})$. There are two major algorithms belonging to that class: The *Hamiltonian* or *hybrid Monte Carlo* (HMC) method [180, 187–189] and the *Metropolis-adjusted Langevin algorithm* (MALA) [189–191], which shall briefly be presented below.

MALA is based on discrete Langevin diffusion dynamics [189, 191] and performs proposals according to

$$
\boldsymbol{\lambda}' = \boldsymbol{\lambda} + \frac{1}{2}\epsilon^2\nabla_{\lambda}\log p(\boldsymbol{\lambda}) + \epsilon\boldsymbol{\xi}, \qquad \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{4.98}
$$

i.e., the proposal density $Prop(\boldsymbol{\lambda}'|\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{\lambda}'|\boldsymbol{\mu}_{MALA}(\boldsymbol{\lambda},\epsilon),\epsilon^2\boldsymbol{I})$ is an isotropic normal distribution with mean $\boldsymbol{\mu}_{MALA}(\boldsymbol{\lambda},\epsilon) = \frac{1}{2}\epsilon^2\nabla_\lambda \log p(\boldsymbol{\lambda})$. The parameter ϵ needs to be prespecified and allows to control the step size of the MALA MCMC algorithm. As the MALA proposal density favors steps along the gradient of $\log p(\boldsymbol{\lambda})$, it is more likely that $p(\boldsymbol{\lambda}') \gtrsim p(\boldsymbol{\lambda})$, i.e., bigger steps can be performed compared to a pure Metropolis random walk leading to shorter integrated autocorrelation times τ_{int}. It has been shown that the optimal acceptance ratio for MALA MCMC is 0.574 and the numerical complexity is $\mathcal{O}(\dim^{1/3}(\boldsymbol{\lambda}))$ as compared to $\mathcal{O}(\dim(\boldsymbol{\lambda}))$ for random walk MH [192], which makes the algorithm particularly useful when the number of stochastic dimensions $\dim(\boldsymbol{\lambda})$ is high.

MCMC methods typically suffer from multimodality in the target distribution $p(\boldsymbol{\lambda})$ [193], since the Markov chain needs to pass low probability regions in order to explore new modes. Techniques like wormhole HMC [194], parallel tempering [195–197], and sequential Monte Carlo (SMC) [188, 198–200] have been developed to overcome this issue.

Sequential Monte Carlo By combining the advantages of importance sampling and MCMC techniques, sequential Monte Carlo methods represent a particularly efficient way to perform inference even under multimodal probability distributions. Instead of sampling from $p(\boldsymbol{\lambda})$ directly, SMC draws samples from a sequence of distributions $p_0(\boldsymbol{\lambda}),\ldots,p_t(\boldsymbol{\lambda}),\ldots,p_T(\boldsymbol{\lambda}) = p(\boldsymbol{\lambda})$ that are easier to handle, e.g., that have shorter mode mixing times. In practice, this can be realized by associating different inverse temperatures β_t to the different distributions like $p_t(\boldsymbol{\lambda}) \propto p^{\beta_t}(\boldsymbol{\lambda})$, with $0 \leq \beta_0 \leq \ldots \leq \beta_t \leq \beta_{t+1} \leq \ldots \leq \beta_T = 1$. The more intermediate distributions are used, the higher the MC accuracy, at the prize of higher computational cost [201].

Given a set of samples $\left\{\boldsymbol{\lambda}_n^{(t)}\right\}_{n=1}^N$, the intermediate distributions are approximated as

$$p_t(\boldsymbol{\lambda}) \propto \sum_{n=1}^{N} m_n^{(t)} \delta_{\boldsymbol{\lambda}_n^{(t)}}(\boldsymbol{\lambda}), \tag{4.99}$$

where $\{m_n^{(t)}\}_{n=1}^N$ is a set of unnormalized, non-negative weights. Expected values are given according to

$$\langle f(\boldsymbol{\lambda})\rangle_{p_t} = \int f(\boldsymbol{\lambda})p_t(\boldsymbol{\lambda})d\boldsymbol{\lambda} \approx \sum_{n=1}^{N} M_n^{(t)} f(\boldsymbol{\lambda}_n^{(t)}), \tag{4.100}$$

where $M_n^{(t)} = m_n^{(t)} / \sum_{n=1}^N m_n^{(t)}$ are the weights after normalization. Starting from a set of samples $\{\boldsymbol{\lambda}_n^{(0)}\}_{n=1}^N$ from the initial distribution $p_0(\boldsymbol{\lambda})$ and uniform weights $M_n^{(0)} = 1/N$, the samples are repeatedly *reweighed*, *resampled*, and *rejuvenated* until the samples are distributed according to $p(\boldsymbol{\lambda})$.

The reweighing step is assigning new weights $m_n^{(t+1)}$ to the particles $\boldsymbol{\lambda}_n^{(t)}$ to correct the difference between $p_t(\boldsymbol{\lambda})$ and $p_{t+1}(\boldsymbol{\lambda})$ via importance sampling, i.e., $m_n^{(t+1)} =$

Algorithm 3 : The sequential Monte Carlo algorithm

```
Input : $t \leftarrow 0, M_n^{(0)} = \frac{1}{N}$,
        and $\{\boldsymbol{\lambda}_n^{(0)}\}_{n=1}^N$ according to $p_0(\boldsymbol{\lambda})$ ;        // Initialization
1  for $t \leftarrow 0$ to $T$ do
2      Reweigh: $m_n^{(t+1)} = m_n^{(t)} \frac{p_{t+1}(\boldsymbol{\lambda}_n^{(t)})}{p_t(\boldsymbol{\lambda}_n^{(t)})}$;
3      Normalize: $M_n^{(t+1)} = \frac{m_n^{(t+1)}}{\sum_{n=1}^N m_n^{(t+1)}}$;
4      $N_{\mathrm{eff}} = \frac{1}{\sum_{n=1}^N (M_n^{(t+1)})^2}$;
5      if $N_{eff} < N_{threshold}$ ;        // $N_{\mathrm{eff}}$ drops below threshold
6      then
7          Resample: $\langle N_n^{(t+2)} \rangle = N M_n^{(t+1)}$;
8      Rejuvenate: perturb $\boldsymbol{\lambda}_n^{(t)} \to \boldsymbol{\lambda}_n^{(t+1)}$ via $Prop(\boldsymbol{\lambda}'|\boldsymbol{\lambda}_n^{(t)})$ and accept
        with Equation (4.94)
9  end
10 return $\boldsymbol{\lambda}^{(0)}, \dots, \boldsymbol{\lambda}^{(t)}, \dots, \boldsymbol{\lambda}^{(N)}$ ;        // Samples from $p(\boldsymbol{\lambda})$
```

$m_n^{(t)} \frac{p_{t+1}(\boldsymbol{\lambda}_n^{(t)})}{p_t(\boldsymbol{\lambda}_n^{(t)})}$.

It is straightforward to show that with N samples and non-uniform weights $M_n^{(t)}$, one would have the same MC error as in Equation (4.86) with an *effective sample size* of $N_{\mathrm{eff}}^{(t)} = 1/\sum_{n=1}^N (M_n^{(t)})^2$, which is N if all $M_n^{(t)} = \frac{1}{N}$ and reduces to 1 if all $M_n^{(t \neq t')} = 0$ except for one $M_n^{(t')} = 1$. Therefore, if the effective sample size N_{eff} drops below a certain threshold, e.g., $N_{\mathrm{eff}} \leq \frac{N}{2}$, then the particles are resampled.

In the resampling step, each particle is copied $N_n^{(t)}$ times s.t. $\sum_{n=1}^N N_n^{(t)} = N$ and $\langle N_n^{(t)} \rangle = N M_n^{(t)}$ [200]. That means that particles with high weights are copied several times, and on the other side, particles of low weight are likely to be discarded. After resampling, every single copy of a sample is assigned the uniform weight $M_n^{(t+1)} = 1/N$.

Finally, the rejuvenation step perturbs the samples $\{\boldsymbol{\lambda}_n^{(t)}\}_{n=1}^N$ using an MCMC transition kernel as discussed in the previous subsections, i.e., new locations are proposed using a suitable proposal density $Prop(\boldsymbol{\lambda}_n'|\boldsymbol{\lambda}_n^{(t)})$ and accepted according to Equation (4.94), assuming the new $p_{t+1}(\boldsymbol{\lambda})$ for the stationary Markov density. The algorithm is summarized in 3.

4.3.4 Variational inference

Just as in the Laplace approximation method discussed in Section 4.3.2, the purpose of *variational inference* (VI) (also called *variational Bayes* or, in physics literature, *variational free energy minimization*) [202–204] is to find an analytically better tractable approximate probability distribution $p_{VI}(\boldsymbol{\lambda})$ to the original density $p(\boldsymbol{\lambda})$. With "better tractable" it is meant that expected values can be computed in closed form and/or direct sampling methods exist (i.e., no MCMC is needed to do Monte Carlo).

Variational inference has its roots in variational mean field theory [205, 206], a popular method to study the physics of quantum and classical many-body systems [207]. To the best of the author's knowledge, mean field theory first found its way into machine learning with [208] and was generalized in [209, 210] to modern variational Bayesian methods.

The basic rationale behind VI is to minimize the *Kullback-Leibler* (KL) *divergence* [163, 164]

$$D_{\mathrm{KL}}(p_{VI}||p) = \int p_{VI}(\boldsymbol{\lambda}) \log\left(\frac{p_{VI}(\boldsymbol{\lambda})}{p(\boldsymbol{\lambda})}\right) d\boldsymbol{\lambda} \tag{4.101}$$

between the original distribution $p(\boldsymbol{\lambda})$ and a *variational approximation* $p_{VI}(\boldsymbol{\lambda})$ out of a family of tractable distributions like, e.g., a set of distributions $p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})$ controlled by some parameters $\boldsymbol{\xi}$. The KL divergence can be viewed as a non-symmetric (i.e., $D_{\mathrm{KL}}(p_{VI}||p) \neq D_{\mathrm{KL}}(p||p_{VI})$) distance measure between the probability distributions p_{VI} and p. It can be shown that $D_{\mathrm{KL}}(p_{VI}||p) \geq 0$ [155], with $D_{\mathrm{KL}}(p_{VI}||p) = 0$ if and only if the two distributions are identical, $p_{VI}(\boldsymbol{\lambda}) = p(\boldsymbol{\lambda})$.

Mean-field approximation

The *variational mean field approximation* is obtained by imposing the independence assumption

$$p_{VI}(\boldsymbol{\lambda}) = \prod_{i=1}^{M} p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]}), \tag{4.102}$$

where $\boldsymbol{\lambda}^{[1]}, \ldots, \boldsymbol{\lambda}^{[M]}$ are subcomponents of $\boldsymbol{\lambda}$ with $\boldsymbol{\lambda}^{[i]} \cap \boldsymbol{\lambda}^{[j]} = \emptyset$ for $i \neq j$ and $\bigcup_{i=1}^{M} \boldsymbol{\lambda}^{[i]} = \boldsymbol{\lambda}$ and then minimizing the functional $D_{KL}(p_{VI}||p)$ w.r.t. $p_{VI}^{[i]}$ under the normalization constraints

$$\int p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]}) d\boldsymbol{\lambda}^{[i]} = 1. \tag{4.103}$$

To do so, Equation (4.102) is plugged in to Equation (4.101) and the normalization constraints Equation (4.103) are added via Lagrange multipliers ζ_i to yield the objective functional

$$\begin{aligned}
\mathcal{J}[p_{VI};\boldsymbol{\zeta}] &= \int \prod_{i=1}^{M} p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]}) \log \frac{\prod_{j=1}^{M} p_{VI}^{[j]}(\boldsymbol{\lambda}^{[j]})}{p(\boldsymbol{\lambda})} d\boldsymbol{\lambda} + \sum_{k=1}^{M} \zeta_k \left(\int p_{VI}^{[k]}(\boldsymbol{\lambda}^{[k]}) d\boldsymbol{\lambda}^{[k]} - 1 \right) \\
&= \sum_{i=1}^{M} \int p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]}) \log p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]}) d\boldsymbol{\lambda}^{[i]} - \int \prod_{i=1}^{M} p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]}) \log p(\boldsymbol{\lambda}) d\boldsymbol{\lambda} \\
&\quad + \sum_{k=1}^{M} \zeta_k \left(\int p_{VI}^{[k]}(\boldsymbol{\lambda}^{[k]}) d\boldsymbol{\lambda}^{[k]} - 1 \right)
\end{aligned} \tag{4.104}$$

which is to be minimized w.r.t. $p_{VI}^{[i]}$. A necessary condition is that the first order variations $\delta\mathcal{J}$ w.r.t. $p_{VI}^{[i]}$ need to vanish, i.e.

$$\begin{aligned}0 &\overset{!}{=} \delta\mathcal{J}(\boldsymbol{\lambda}^{[i]}, \boldsymbol{\zeta}^{[i]}) = \\ &= \int \delta p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]}) \left(\log p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]}) - \int \prod_{j \neq i} p_{VI}^{[j]}(\boldsymbol{\lambda}^{[j]}) \log p(\boldsymbol{\lambda}) d\boldsymbol{\lambda}^{[-i]} + \boldsymbol{\zeta}^{[i]} - 1 \right) d\boldsymbol{\lambda}^{[i]},\end{aligned} \tag{4.105}$$

where $d\boldsymbol{\lambda}^{[-i]} = \prod_{j \neq i} d\boldsymbol{\lambda}^{[j]}$. As this must hold for arbitrary variations $\delta p_{VI}^{[i]}$,

$$\log p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]}) - \int \prod_{j \neq i} p_{VI}^{[j]}(\boldsymbol{\lambda}^{[j]}) \log p(\boldsymbol{\lambda}) d\boldsymbol{\lambda}^{[-i]} + \boldsymbol{\zeta}^{[i]} - 1 = 0, \tag{4.106}$$

or

$$\log p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]}) = \langle \log p(\boldsymbol{\lambda}) \rangle_{j \neq i} - \boldsymbol{\zeta}^{[i]} + 1 \tag{4.107}$$

where $\langle \,.\, \rangle_{i \neq k}$ denotes expectations w.r.t. $\prod_{j \neq i} p_{VI}^{[j]}(\boldsymbol{\lambda}^{[j]})$. The normalization constraints Equation (4.103) are recovered using the condition that $\nabla_{\boldsymbol{\zeta}}\mathcal{J} \overset{!}{=} 0$ and is fixing the additive constant $\boldsymbol{\zeta}^{[i]} - 1$ in Equation (4.107). Finally, taking the exponential on both sides, it is found that

$$p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]}) = \frac{\exp \langle \log p(\boldsymbol{\lambda}) \rangle_{j \neq i}}{\int \exp \langle \log p(\boldsymbol{\lambda}) \rangle_{j \neq i} d\boldsymbol{\lambda}^{[i]}}. \tag{4.108}$$

It is noted that Equation (4.108) only gives an implicit solution for $p_{VI}^{[i]}(\boldsymbol{\lambda}^{[i]})$, since $\langle \log p(\boldsymbol{\lambda}) \rangle_{j \neq i}$ depends on all other $p_{VI}^{[j]}(\boldsymbol{\lambda}^{[j]})$. To get an explicit solution, it is therefore mandatory to repeatedly cycle over all j until convergence.

Stochastic variational inference

In *stochastic variational inference* [211–213], the approximate variational distribution $p_{VI}(\boldsymbol{\lambda})$ is assumed to be the member of a parametric family of distributions $p_{VI}(\boldsymbol{\lambda}) = p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})$ parametrized by $\boldsymbol{\xi}$. A classic example would be the family of multivariate normal distributions with unknown mean $\boldsymbol{\mu}_{VI}$ and covariance $\boldsymbol{\Sigma}_{VI}$, i.e., $\boldsymbol{\xi} = \binom{\boldsymbol{\mu}_{VI}}{\boldsymbol{\Sigma}_{VI}}$.
Again, the objective is to minimize the KL divergence $D_{\mathrm{KL}}(p_{VI}||p)$ as given in Equation (4.101),

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}} \int p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi}) \log \left(\frac{p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})}{p(\boldsymbol{\lambda})} \right) d\boldsymbol{\lambda}, \tag{4.109}$$

and then use $p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi}^*)$ as an easily tractable approximation to the original distribution $p(\boldsymbol{\lambda})$.

To efficiently optimize Equation (4.109), gradients w.r.t. the variational parameters $\boldsymbol{\xi}$ should be used. Using $\nabla_{\boldsymbol{\xi}}[p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})] = p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})\nabla_{\boldsymbol{\xi}}[\log p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})]$ and

$\int \nabla_{\boldsymbol{\xi}}[p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})]d\boldsymbol{\lambda} = 0$, it is found that

$$\begin{aligned} \nabla_{\boldsymbol{\xi}}[D_{\mathrm{KL}}(p_{VI}||p)] &= \int \nabla_{\boldsymbol{\xi}}[p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})] \log\left(\frac{p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})}{p(\boldsymbol{\lambda})}\right) d\boldsymbol{\lambda} \\ &+ \int p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi}) \nabla_{\boldsymbol{\xi}}[\log p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})]d\boldsymbol{\lambda} \\ &= \left\langle \nabla_{\boldsymbol{\xi}}[\log p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})] \left(\log p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi}) - \log p(\boldsymbol{\lambda})\right)\right\rangle_{p_{VI}} . \end{aligned} \tag{4.110}$$

Depending on the specific form of p_{VI} and p, some terms in Equation (4.110) may be computed analytically. In general, Equation (4.110) can be estimated via Monte Carlo as

$$\nabla_{\boldsymbol{\xi}}[D_{\mathrm{KL}}(p_{VI}||p)] \approx \frac{1}{N}\sum_{n=1}^{N} \nabla_{\boldsymbol{\xi}}[\log p_{VI}(\boldsymbol{\lambda}^{(n)}|\boldsymbol{\xi})] \left(\log p_{VI}(\boldsymbol{\lambda}^{(n)}|\boldsymbol{\xi}) - \log p(\boldsymbol{\lambda}^{(n)})\right), \tag{4.111}$$

where $\boldsymbol{\lambda}^{(n)} \sim p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})$. Unfortunately, this gradient estimator tends to have very high variance [100, 211], i.e., many potentially expensive samples are needed to obtain decent estimates.

A prominent strategy to reduce Monte Carlo noise in Equation (4.111) is to apply the *reparametrization trick* [100, 214]. We define a differentiable transformation $\boldsymbol{g}$

$$\boldsymbol{\lambda} = \boldsymbol{g}(\boldsymbol{\epsilon}, \boldsymbol{\xi}) \tag{4.112}$$

with an auxiliary noise variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. Since $p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi}) \prod_i d\lambda_i = p(\boldsymbol{\epsilon}) \prod_j d\epsilon_j$, $D_{\mathrm{KL}}(p_{VI}||p)$ can be rewritten as

$$D_{\mathrm{KL}}(p_{VI}||p) = \langle \log p_{VI}(\boldsymbol{g}(\boldsymbol{\epsilon},\boldsymbol{\xi})|\boldsymbol{\xi})\rangle_{p(\boldsymbol{\epsilon})} - \langle \log p(\boldsymbol{g}(\boldsymbol{\epsilon},\boldsymbol{\xi}))\rangle_{p(\boldsymbol{\epsilon})} \tag{4.113}$$

and the gradient

$$\nabla_{\boldsymbol{\xi}}[D_{\mathrm{KL}}(p_{VI}||p)] = \left\langle \nabla_{\boldsymbol{\xi}}[\log p_{VI}(\boldsymbol{g}(\boldsymbol{\epsilon},\boldsymbol{\xi})|\boldsymbol{\xi})]\right\rangle_{p(\boldsymbol{\epsilon})} - \left\langle \nabla_{\boldsymbol{g}}[\log p(\boldsymbol{g}(\boldsymbol{\epsilon},\boldsymbol{\xi}))] \cdot \nabla_{\boldsymbol{\xi}}[\boldsymbol{g}]\right\rangle_{p(\boldsymbol{\epsilon})} \tag{4.114}$$

which exhibits considerably lower Monte Carlo noise compared to Equation (4.111) since it is incorporating gradient information $\nabla_{\boldsymbol{g}} p(\boldsymbol{g}(\boldsymbol{\epsilon},\boldsymbol{\xi}))$ of the original distribution $p(\boldsymbol{\lambda})$. The gradient $\nabla_{\boldsymbol{\xi}}[D_{\mathrm{KL}}(p_{VI}||p)]$ is then passed to a suitable gradient-based stochastic optimization algorithm [171–173], see Section 4.4. It is noted though that the computation of such gradients can be computationally expensive or even impossible which is the case, e.g., when $p(\boldsymbol{\lambda})$ is a posterior that contains a likelihood function involving a complex computer simulation.

Finally, given the approximation $p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi}^*) \approx p(\boldsymbol{\lambda})$, the expectation value from Equation (4.67) can be approximated as

$$\langle f(\boldsymbol{\lambda})\rangle = \int f(\boldsymbol{\lambda}) p_{VI}(\boldsymbol{\lambda}) d\boldsymbol{\lambda}, \tag{4.115}$$

which, given a smart choice of the parametric family of distributions $p_{VI}(\boldsymbol{\lambda}|\boldsymbol{\xi})$, can be solved in closed form or approximated via direct Monte Carlo.

4.4 Stochastic gradient ascent

Stochastic gradient ascent [215–217] (SGA, also called *stochastic gradient descent* (SGD) in minimization problems) algorithms are among the most popular stochastic optimization algorithms and their recent progressions are a central building block of the present-day success of machine learning and artificial intelligence methods[7] in various fields of science and industry. SGA is a method to maximize an objective function $\mathcal{J}(\boldsymbol{\xi})$ that is only known by noisy estimates of the gradient $\nabla_{\boldsymbol{\xi}}\mathcal{J}(\boldsymbol{\xi})$,

$$\nabla_{\boldsymbol{\xi}}\mathcal{J}(\boldsymbol{\xi}) = \nabla_{\boldsymbol{\xi}} \int \tilde{\mathcal{J}}(\boldsymbol{\xi},\boldsymbol{\lambda})p(\boldsymbol{\lambda})d\boldsymbol{\lambda} \approx \frac{1}{N}\sum_{n=1}^{N} \nabla_{\boldsymbol{\xi}}\tilde{\mathcal{J}}(\boldsymbol{\xi},\boldsymbol{\lambda}^{(n)}), \qquad \boldsymbol{\lambda}^{(n)} \sim p(\boldsymbol{\lambda}). \tag{4.116}$$

For ease of notation, we will denote the mean gradient estimate

$$\boldsymbol{g}^{(t)} = \frac{1}{N}\sum_{n=1}^{N} \nabla_{\boldsymbol{\xi}}\tilde{\mathcal{J}}(\boldsymbol{\xi}^{(t)},\boldsymbol{\lambda}^{(n)}) \approx \langle \nabla_{\boldsymbol{\xi}}\mathcal{J}(\boldsymbol{\xi})\rangle \tag{4.117}$$

in the following. The basic principle of SGA is to update the parameters $\boldsymbol{\xi}$ by steps along a preferred direction $\boldsymbol{m}^{(t)} = \boldsymbol{m}^{(t)}(\boldsymbol{g}^{(0)},\dots,\boldsymbol{g}^{(t)})$ which is, in general, a function of previous gradient evaluations $\boldsymbol{g}^{(0)},\dots,\boldsymbol{g}^{(t)}$. In more mathematical terms,

$$\boldsymbol{\xi}^{(t+1)} = \boldsymbol{\xi}^{(t)} + \boldsymbol{\eta}^{(t)} \circ \boldsymbol{m}^{(t)} \tag{4.118}$$

where $\circ$ denotes element-wise multiplication, $\boldsymbol{\eta}^{(t)}$ is the *learning rate* and $\boldsymbol{\xi}^{(0)}$ needs to be set for initialization. The recursion relation defined by Equation (4.118) is repeatedly evaluated until a suitable convergence criterion is fulfilled, e.g., if $|\boldsymbol{g}^{(t)}|^2 < \ell^2$, with a user-specified threshold ℓ.

4.4.1 The Robbins-Monro algorithm

The Robbins-Monro algorithm [215] uses

$$\boldsymbol{m}^{(t)} = \boldsymbol{g}^{(t)} \tag{4.119}$$

and is guaranteed to converge to a local maximum if for the learning rate

$$\sum_{t=1}^{\infty} \eta_i^{(t)} = \infty, \qquad \sum_{t=1}^{\infty} (\eta_i^{(t)})^2 < 0, \qquad \eta_i^{(t)} > 0. \tag{4.120}$$

[7]See, e.g., the documentation of the deep learning libraries TensorFlow and PyTorch on neural network training.

Typical choices for $\eta_i^{(t)}$ can be constructed using the Riemann ζ-function $\zeta(s) = \sum_{t=1}^{\infty} t^{-s}$ which is convergent for $\mathrm{Re}(s) > 1$ [218]. Thus, one may set

$$\eta_i^{(t)} = \frac{\alpha_i}{\beta_i + t^{\gamma_i}} \tag{4.121}$$

with some adjustable parameters $\alpha_i, \beta_i > 0$ and $\gamma_i \in \left(\frac{1}{2}, 1\right]$.

Although convergence to a local optimum is ensured, there are a few shortcomings using the above algorithm which directly affect its computational efficiency. First, choosing proper learning rate parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ can be difficult, especially in high dimensions $\dim(\boldsymbol{\xi})$. Using shared values $\alpha_i = \alpha, \beta_i = \beta, \gamma_i = \gamma$ for all dimensions is possible, but leads to slow convergence especially if the dimensions have very different length scales. Moreover, valuable information from previous gradient evaluations $\boldsymbol{g}^{(t-1)}, \ldots, \boldsymbol{g}^{(0)}$ is discarded, which could be used to reduce estimation errors on the true gradient $\nabla_{\boldsymbol{\xi}} \mathcal{J}(\boldsymbol{\xi})$ or to incorporate curvature information to the update equation.

4.4.2 AdaGrad

The AdaGrad algorithm [219] adjusts the learning rate $\boldsymbol{\eta}$ automatically according to the different length scales of the inputs ξ_i. It does so by modifying the learning rate to be

$$\eta_i^{(t)} = \frac{\eta_0}{\sqrt{\bar{G}_i^{(t)} + \epsilon}}, \tag{4.122}$$

where

$$\bar{G}_i^{(t)} = \sum_{t'=0}^{t} (g_i^{(t')})^2, \tag{4.123}$$

i.e., the learning rate η_0 is divided by square root of the sum of squared derivatives $(g_i^{(t')})^2$ up to iteration t ($\epsilon > 0$ is a regularity parameter that avoids division by 0). This is guided by the following intuition: if the sum of squared gradient in a certain direction ξ_i is much higher than the others, one can assume that the objective function $\mathcal{J}(\boldsymbol{\xi})$ is strongly oscillating along ξ_i and it would be reasonable to do smaller steps in that direction. Otherwise, if the sum of squared gradients along ξ_i is low, the objective function $\mathcal{J}(\boldsymbol{\xi})$ is rather smooth and it is safe to do larger update steps.

As for the Robbins-Monro algorithm in Section 4.4.1, the update equation Equation (4.118) holds and $\boldsymbol{m}^{(t)} = \boldsymbol{g}^{(t)}$ is used.

4.4.3 RMSprop

As the AdaGrad algorithm is dividing the learning rate η_0 summing up all squared derivatives up to the current iteration t, the step sizes are dropping overly quickly and the optimization may stall before the maximum $\boldsymbol{\xi}^*$ is reached. RMSprop[8] fixes this issue by not taking the sum over all the gradients up to iteration t, but using a moving average of the form

$$G_i^{(t)} = \beta G_i^{(t-1)} + (1-\beta)(g_i^{(t)})^2. \tag{4.124}$$

A value of $\beta = 0.9$ is suggested by the inventor G. Hinton. The learning rate becomes

$$\eta_i^{(t)} = \frac{\eta_0}{\sqrt{G_i^{(t)}} + \epsilon}. \tag{4.125}$$

Apart from that, RMSprop is identical to the Robbins-Monro and AdaDelta algorithms.

4.4.4 ADAM

The currently probably most popular stochastic gradient ascent algorithm is *adaptive moment estimation* (ADAM) [220]. ADAM not only estimates the second moment of the gradient $G_i^{(t)}$ as in Equation (4.124), but also the mean of the gradient $M_i^{(t)}$ (first moment), thus the name of the method. The exponentially decaying moment estimates read

$$\begin{aligned} M_i^{(t)} &= \alpha M_i^{(t-1)} + (1-\alpha) g_i^{(t)}, \\ G_i^{(t)} &= \beta G_i^{(t-1)} + (1-\beta)(g_i^{(t)})^2. \end{aligned} \tag{4.126}$$

The quantity $M_i^{(t)}$ transfers knowledge of the gradient from previous iterations to the current iteration t and can therefore be understood as a measure for 'momentum' in the optimization process.

As both moment estimates are initialized as $M_i^{(t=0)} = 0, G_i^{(t=0)} = 0$, the moving averages Equation (4.126) are biased towards smaller values in magnitude. This bias is counteracted in the original paper [220] by setting[9]

$$\begin{aligned} \hat{M}_i^{(t)} &= \frac{M_i^{(t)}}{1-\alpha^t}, \\ \hat{G}_i^{(t)} &= \frac{G_i^{(t)}}{1-\beta^t}. \end{aligned} \tag{4.127}$$

Finally, ADAM uses the update Equation (4.118) with step direction $\boldsymbol{m}^{(t)} = \hat{\boldsymbol{M}}^{(t)}$ and the learning rate

$$\eta_i^{(t)} = \frac{\eta_0}{\sqrt{\hat{G}_i^{(t)}} + \epsilon}. \tag{4.128}$$

[8]unpublished, see this lecture by G. Hinton:
http://www.cs.toronto.edu/ hinton/coursera/lecture6/lec6.pdf

[9]It is noted that α^t, β^t means 'α, β to the power of t', i.e., t is not to be confused with an index here.

For the parameters $\alpha, \beta \in [0,1)$, η_0 and ϵ the authors suggest the default parameters [220] $\alpha = 0.9, \beta = 0.999, \epsilon = 10^{-8}$ and $\eta_0 = 0.001$.

4.5 Surrogate modeling

Many-query applications of complex computer codes such as inverse problems, design/optimization or general uncertainty propagation problems typically require a good many of forward model evaluations to explore the parameter space. Many computer simulations of scientific or engineering problems take up to several days of runtime even on modern hardware, which is prohibitive for many of the above problems. Given a limited computational budget, it is thus of paramount interest to extract as much information as possible from a limited dataset of forward model runs $\mathcal{D} = \left\{ \boldsymbol{\lambda}^{(n)}, \boldsymbol{u}_f \boldsymbol{\lambda}^{(n)}) \right\}_{n=1}^{N}$.
A common approach is to construct, based on the set $\mathcal{D}$ of forward evaluations, a potentially much cheaper surrogate model at the expense of model accuracy. This accuracy typically improves with the size of the dataset $\mathcal{D}$, but convergence to the true forward model is usually not given. The task of finding a good surrogate based on limited data $\mathcal{D}$ is very similar to a supervised learning problem. Off-the-shelf machine learning algorithms such as neural networks (Section 4.5.1) or Gaussian processes (Section 4.5.2) have been applied also in the context of SPDEs, see e.g. [46, 49, 55, 72]. With the (generalized) polynomial chaos expansion and the reduced basis method, two SPDE-specific classes of surrogate models are presented in Section 4.5.3 and Section 4.5.4.

However, in the field of random heterogeneous media, the number of parameters $\dim(\boldsymbol{\lambda})$ needed to accurately describe the exact topology of a sample microstructure is typically very high. As we want to conduct numerical experiments under random variation of microstructures, it is that very $\boldsymbol{\lambda}$ which plays the role of the input that, when plugged in to the surrogate model, should give an estimate of the true model response $\boldsymbol{u}_f(\boldsymbol{\lambda})$. The high dimension of $\boldsymbol{\lambda}$ is seriously impeding the task of finding an accurate surrogate model – a symptom of the *curse of dimensionality* [155], a term introduced by Richard Bellman in 1957 [221]. It states that the average distance of sample points increases exponentially with dimension which exacerbates high-dimensional regression.

This is a central motivation for the development of the physics-aware surrogate presented in Chapter 5 which, by exclusion of a myriad of potential model outputs violating essential physical principles, generally scales much better with the stochastic input dimension $\dim(\boldsymbol{\lambda})$ than the models mentioned above.

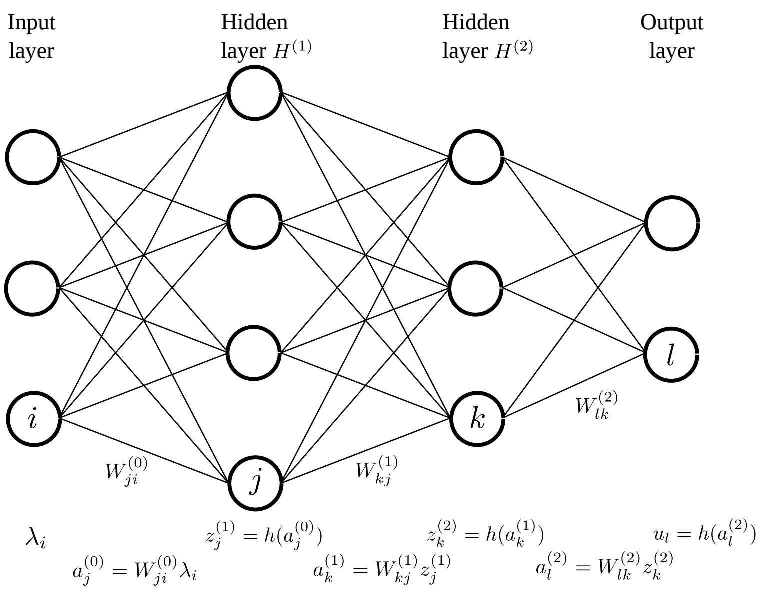

Input layer
Hidden layer $H^{(1)}$
Hidden layer $H^{(2)}$
Output layer
i
j
k
l
$W_{ji}^{(0)}$
$W_{kj}^{(1)}$
$W_{lk}^{(2)}$
λ_i
$a_j^{(0)} = W_{ji}^{(0)}\lambda_i$
$z_j^{(1)} = h(a_j^{(0)})$
$a_k^{(1)} = W_{kj}^{(1)} z_j^{(1)}$
$z_k^{(2)} = h(a_k^{(1)})$
$a_l^{(2)} = W_{lk}^{(2)} z_k^{(2)}$
$u_l = h(a_l^{(2)})$

FIGURE 4.8: Schematic representation of a feed-forward neural network with two hidden layers. The model function $\hat{\boldsymbol{f}}(\boldsymbol{\lambda}, \hat{\boldsymbol{\theta}})$ consists of successive linear combinations of the nodal values $\boldsymbol{a}^{(m)} = \boldsymbol{W}^{(m)}\boldsymbol{z}^{(m)}$, and subsequently passing them through a nonlinear activation function, $z_j^{(m+1)} = h(a_j^{(m+1)})$. Gradients w.r.t. the parameters $\hat{\boldsymbol{\theta}} = \left\{\boldsymbol{W}^{(m)}\right\}_{m=0}^{M}$ ('weights', depicted by the connections between the neurons) can efficiently be computed using the chain rule (called backpropagation or backprop, see [78, 223, 224]).

4.5.1 Artificial neural networks

The linear models that were discussed in Section 4.2 to introduce different kinds of prior models proved to be valuable in that many computational steps in training and prediction were doable in closed form. However, their regression quality is strongly dependent on a good choice of basis/feature functions $\boldsymbol{\varphi}(\lambda)$. This is a desirable property in settings where expressive features for the prediction of the regression targets $\boldsymbol{u}$ are known a priori.

Regularly, such features are not given beforehand and one is better of with more generic, of-the-shelf models $\hat{f}(\lambda, \hat{\boldsymbol{\theta}})$ that do not rely on a priori information about the data. The perhaps most popular class of such models in modern-day machine learning are *artificial* or *deep neural networks* (ANNs/DNNs) [63, 64, 222], which shall be introduced in this subsection.

The most basic form of an ANN is the *feed-forward network* or *multilayer perceptron* (MLP) which is depicted schematically in Figure 4.8. The original intention behind using a neural network model architecture was to mimic the functionality of the human brain and dates back to [225]. An ANN consists of several layers[10]: the *input layer* which is identical to an input data sample $\boldsymbol{\lambda}^{(n)}$, some *hidden layers* $H^{(m)}$, and

[10]If there is more than one hidden layer, ANNs are commonly referred to as 'deep', i.e. *deep neural networks* (DNNs) [222, 226].

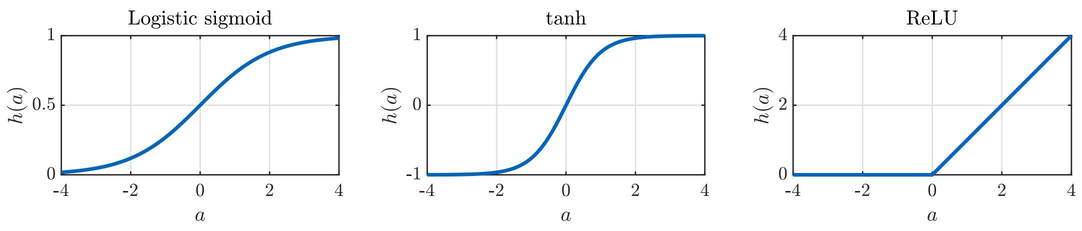


FIGURE 4.9: Some popular activation functions $h(a)$.

the output layer corresponding to the network prediction $\hat{\boldsymbol{u}}^{(n)} = \hat{\boldsymbol{f}}(\boldsymbol{\lambda}^{(n)}, \hat{\boldsymbol{\theta}})$. The layers themselves consist of different numbers of *neurons* $\boldsymbol{z}^{(m)}$ which are linearly combined by the network *weights* $\boldsymbol{W}^{(m)}$ and passed through an *activation function* $h(\cdot)$ to yield the neuron values $\boldsymbol{z}^{(m+1)}$ of the subsequent layer. Hence, the model function $\hat{\boldsymbol{u}} = \hat{\boldsymbol{f}}(\boldsymbol{\lambda}, \hat{\boldsymbol{\theta}})$ of a feed-forward network can be summarized as

$$\begin{aligned} z_j^{(1)} &= h\left(\boldsymbol{W}_{ji}^{(0)} \lambda_i\right) \\ &\vdots \\ z_j^{(m+1)} &= h\left(\boldsymbol{W}_{ji}^{(m)} z_i^{(m)}\right) \\ &\vdots \\ \hat{u}_j &= h\left(\boldsymbol{W}_{ji}^{(M)} z_i^{(M)}\right), \end{aligned} \tag{4.129}$$

where summation over repeated indices is implied and the weights $\hat{\boldsymbol{\theta}} = \left\{\boldsymbol{W}^{(m)}\right\}_{m=1}^{M}$ are model parameters to be found by training. Some common activation functions $h(\cdot)$ are shown in Figure 4.9. In principle, $h(\cdot)$ can be an arbitrary function and can differ from neuron to neuron. However, there are some desirable properties, e.g., it should be nonlinear [227], differentiable, and monotonic [228].

ANNs are trained by minimization of a suitable loss function. For regression problems, the most common is the ℓ_2 error

$$\hat{\boldsymbol{\theta}}^* = \arg\min_{\hat{\boldsymbol{\theta}}} \sum_{n=1}^{N} \|\boldsymbol{u}^{(n)} - \hat{\boldsymbol{f}}(\boldsymbol{\lambda}^{(n)}, \hat{\boldsymbol{\theta}})\|^2 \tag{4.130}$$

which can be minimized using a (stochastic) gradient ascent algorithm (see Section 4.4). For performance reasons, usually only a small subset of terms in the sum of Equation (4.130) are sampled to get stochastic gradient estimates w.r.t. the parameters $\hat{\boldsymbol{\theta}}$ (mini-batch gradient descent).

For a single term n, the gradient w.r.t. the weights $W_{jk}^{(M-m)}$ is given as

$$\begin{aligned}\frac{\partial}{\partial W_{jk}^{(M-m)}}\|\boldsymbol{u}^{(n)} - \hat{\boldsymbol{f}}(\boldsymbol{\lambda}^{(n)}, \hat{\boldsymbol{\theta}})\|^2 &= \sum_{i_1}(u_{i_1}^{(n)} - \hat{f}_{i_1})\frac{\partial \hat{f}_{i_1}}{\partial W_{jk}^{(M-m)}} \\ = \sum_{i_1}(u_{i_1}^{(n)} - \hat{f}_{i_1})h'(a_{i_1}^{(M)})\sum_{i_2} W_{i_1 i_2}^{(M)} h'(a_{i_2}^{(M-1)}) &\sum_{i_3} W_{i_2 i_3}^{(M-1)} h'(a_{i_3}^{(M-2)}) \cdot \ldots \\ \ldots \cdot W_{i_m j}^{(M-m+1)} h'(a_j^{(M-m)}) z_k^{(M-m)}&. \end{aligned} \tag{4.131}$$

Denoting

$$\begin{aligned}\Delta_j^{(M)} &= \sum_i (u_i^{(n)} - \hat{f}_i)h'(a_i^{(M)})W_{ij}^{(M)}, \\ \Delta_j^{(M-m)} &= \sum_i \Delta_i^{(M-m+1)} W_{ij}^{(M-m)} h'(a_i^{(M-m)}) \quad \text{for} \quad m \geq 1,\end{aligned} \tag{4.132}$$

the above derivative can be written as

$$\frac{\partial}{\partial W_{jk}^{(M-m)}}\|\boldsymbol{u}^{(n)} - \hat{\boldsymbol{f}}(\boldsymbol{\lambda}^{(n)}, \hat{\boldsymbol{\theta}})\|^2 = \Delta_j^{(M-m+1)} h'(a_j^{(M-m)}) z_k^{(M-m)}. \tag{4.133}$$

It is noted that to compute $\Delta_j^{(M-m+1)}$, it is necessary to start at the output layer and evaluate $\Delta_j^{(M)}$, then successively compute to $\Delta_j^{(M-1)}, \Delta_j^{(M-2)}, \ldots$ up to $\Delta_j^{(M-m+1)}$. This process starts with the error term $\Delta_j^{(M)}$ at the output end of the network and propagates errors backwards to the layer $(M - m + 1)$ and is therefore called *error backpropagation* or *backprop* (BP) for short [78, 223, 224]). With the current parameter estimates $\hat{\boldsymbol{\theta}} = \left\{\boldsymbol{W}^{(m)}\right\}_{m=0}^{M}$, the input $\boldsymbol{\lambda}^{(n)}$ can be forward propagated through the network to precompute all neuron states $a_i^{(m)}$ and activations $z_i^{(m+1)}$. Thereupon, they can be used to compute all $\Delta_j^{(m)}$ according to Equation (4.132) and finally evaluate the gradient w.r.t. $\boldsymbol{W}^{(M-m)}$ making use of Equation (4.133). The possibility to efficiently compute gradients w.r.t. the model parameters $\hat{\boldsymbol{\theta}}$ using backpropagation is one of the main reasons of the popularity of ANN models in modern day machine learning.

Although a large variety of regularization techniques for ANNs exist [229–232], because of the large number of free parameters $\dim(\hat{\boldsymbol{\theta}})$, they are typically used in problems where big datasets are available. A popular, regularized generalization of the multilayer perceptron that takes advantage of $2D$ input data as given in, e.g., computer vision is the *convolutional neural network* (CNN) [65, 67]. In addition to the fully connected layers of the MLP discussed above, a CNN can also comprise network layers that perform different operations such as convolutions, pooling operations, batch normalization, or activation operations.

In a convolutional layer, convolutions of the output of the previous layer w.r.t. several fixed-size convolutional kernels that are learned from the data are carried out.

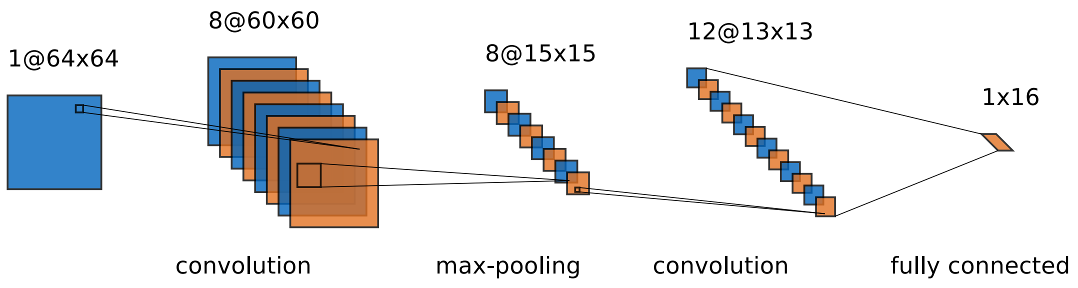


FIGURE 4.10: Schematic representation of the architecture of a convolutional neural network (CNN) for a regression task from 64×64 images $\boldsymbol{\lambda}$ to a 1×16-dimensional output vector $\boldsymbol{u}$. Apart from fully connected layers, layers performing different operations, such as convolutions or pooling, can be applied.

Convolutions on a $2D$ input image $\boldsymbol{\lambda}$ are capable to extract features like edges or simple shapes.

The main purpose of a pooling layer is down-sampling to reduce the number of free parameters of the network. For instance, a max-pooling layer divides the output of the previous layer in non-overlapping rectangular windows and passes the maximum pixel value of each window as an output.

A batch normalization layer shifts and scales the output of the previous layer with shift and scale parameters learned from the data.

A ReLU layer passes all output pixel values from the previous layer through the ReLU activation function, see the right part of Figure 4.9.

The architecture of a simple CNN is represented exemplary in Figure 4.10[11].

4.5.2 Gaussian process regression

A *Gaussian process* [47, 48] (GP) can be viewed as a distribution over functions

$$\hat{f}(\boldsymbol{\lambda}) \sim \mathrm{GP}(m(\boldsymbol{\lambda}), k(\boldsymbol{\lambda}, \boldsymbol{\lambda}')) \tag{4.134}$$

with mean and covariance (or kernel) function $m(\boldsymbol{\lambda}), k(\boldsymbol{\lambda}, \boldsymbol{\lambda}')$ such that

$$\begin{aligned} \left\langle \hat{f}(\boldsymbol{\lambda}) \right\rangle &= m(\boldsymbol{\lambda}), \\ \left\langle \left(\hat{f}(\boldsymbol{\lambda}) - m(\boldsymbol{\lambda})\right) \left(\hat{f}(\boldsymbol{\lambda}') - m(\boldsymbol{\lambda}')\right) \right\rangle &= k(\boldsymbol{\lambda}, \boldsymbol{\lambda}'). \end{aligned} \tag{4.135}$$

A simple example for a Gaussian Process is the parametric linear model given by Equation (4.10) discussed earlier: Given a normal prior $p_0(\hat{\boldsymbol{\theta}}) = \mathcal{N}(\hat{\boldsymbol{\theta}} | \boldsymbol{\mu}_{\hat{\theta}}, \boldsymbol{\Sigma}_{\hat{\theta}})$, the above expected values are

$$m(\boldsymbol{\lambda}) = \boldsymbol{\mu}_{\hat{\theta}}^T \boldsymbol{\varphi}(\boldsymbol{\lambda}), \qquad k(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = \boldsymbol{\varphi}^T(\boldsymbol{\lambda}) \boldsymbol{\Sigma}_{\hat{\theta}} \boldsymbol{\varphi}(\boldsymbol{\lambda}'). \tag{4.136}$$

[11]Generated with http://alexlenail.me/NN-SVG/LeNet.html

Independent from the specific form of the model function $\hat{f}(\boldsymbol{\lambda})$, Equation (4.135) implies that, given a set of inputs $\mathcal{D}_{\boldsymbol{\lambda}} = \left\{\boldsymbol{\lambda}^{(n)}\right\}_{n=1}^{N}$, the model outputs $\hat{\boldsymbol{f}}, \hat{f}_n = \hat{f}(\boldsymbol{\lambda}^{(n)})$, are jointly Gaussian with

$$p(\hat{\boldsymbol{f}}|\mathcal{D}_{\boldsymbol{\lambda}}) = \mathcal{N}(\hat{\boldsymbol{f}}|\boldsymbol{m}, \boldsymbol{C}), \tag{4.137}$$

where $m_n = m(\boldsymbol{\lambda}^{(n)}), C_{mn} = k(\boldsymbol{\lambda}^{(m)}, \boldsymbol{\lambda}^{(n)})$. This in turn suggests that, instead of using a parametric model of the form $\hat{f}(\boldsymbol{\lambda}) = \hat{\boldsymbol{\theta}}^T\boldsymbol{\varphi}(\boldsymbol{\lambda})$, one could directly come up a parametric form for the mean and covariance functions $m(\boldsymbol{\lambda}) = m(\boldsymbol{\lambda}, \boldsymbol{\theta}_m), k(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = k(\boldsymbol{\lambda}, \boldsymbol{\lambda}', \boldsymbol{\theta}_k)$, and then find the optimal parameters $\boldsymbol{\theta}_m, \boldsymbol{\theta}_k$ by maximizing

$$\boldsymbol{\theta}_m^*, \boldsymbol{\theta}_k^* = \arg\max_{\boldsymbol{\theta}_m, \boldsymbol{\theta}_k} \log \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}(\boldsymbol{\theta}_m), \boldsymbol{C}(\boldsymbol{\theta}_k)). \tag{4.138}$$

where $\boldsymbol{u}$ is the vector of output data. For simplicity, it is common to use a zero-mean Gaussian process prior and set $m(\boldsymbol{\lambda}) = 0$. The above log (marginal) likelihood then becomes

$$\log \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}(\boldsymbol{\theta}_m), \boldsymbol{C}(\boldsymbol{\theta}_k)) \propto -\frac{1}{2}\log|\boldsymbol{C}(\boldsymbol{\theta}_k)| - \frac{1}{2}\boldsymbol{u}^T\boldsymbol{C}(\boldsymbol{\theta}_k)\boldsymbol{u} \tag{4.139}$$

and the training process consists of

$$\boldsymbol{\theta}_k^* = \arg\max_{\boldsymbol{\theta}_k} \left(-\frac{1}{2}\log|\boldsymbol{C}(\boldsymbol{\theta}_k)| - \frac{1}{2}\boldsymbol{u}^T\boldsymbol{C}(\boldsymbol{\theta}_k)\boldsymbol{u}\right). \tag{4.140}$$

Just like for finite-dimensional, multivariate normal distributions, the kernel function of a Gaussian process needs to be positive semidefinite, i.e. it needs to fulfill the inequality

$$\int f(\boldsymbol{\lambda})k(\boldsymbol{\lambda}, \boldsymbol{\lambda}', \boldsymbol{\theta}_k)f(\boldsymbol{\lambda}')d\boldsymbol{\lambda}d\boldsymbol{\lambda}' \geq 0 \qquad \forall f \in L_2 \tag{4.141}$$

which directly implies that also the data covariance matrix $\boldsymbol{C}(\boldsymbol{\theta}_k)$ is positive semidefinite. Popular covariance functions are e.g.:

- Squared exponential: $k_{SE}(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = \exp\left(-\frac{1}{2}\sum_i \frac{(\lambda_i - \lambda_i')^2}{\ell_i^2}\right)$;
- Ornstein-Uhlenbeck: $k_{OU}(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = \exp\left(-\sum_i \frac{|\lambda_i - \lambda_i'|}{\ell_i}\right)$;
- Matérn: $k_{\text{Matérn}}(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\sum_i \frac{|\lambda' - \lambda_i'|}{l_i}\right)^{\nu} I_\nu\left(\sqrt{2\nu}\sum_i \frac{|\lambda_i - \lambda_i'|}{\ell_i}\right)$;

where $\ell_i > 0$ is a length scale parameter for dimension λ_i, $\nu > 0$ is a smoothness parameter and I_ν is the modified Bessel function (of second kind) of order ν. Samples from a zero-mean Gaussian process prior with the above covariance functions are shown in the top row of Figure 4.11.

Given the evidence as in Equation (4.137) (with $\boldsymbol{m} = 0$) and the optimal kernel function parameters $\boldsymbol{\theta}_k$ found by Equation (4.140), predictive estimates u_q at $\boldsymbol{\lambda}_q$ are obtained by expanding

$$p(\left(\begin{smallmatrix}\hat{\boldsymbol{f}} \\ u_q\end{smallmatrix}\right)|\mathcal{D}_{\boldsymbol{\lambda}}, \boldsymbol{\lambda}_q) = \mathcal{N}(\left(\begin{smallmatrix}\hat{\boldsymbol{f}} \\ u_q\end{smallmatrix}\right)|\boldsymbol{0}, \bar{\boldsymbol{C}}(\boldsymbol{\theta}_k^*)), \tag{4.142}$$

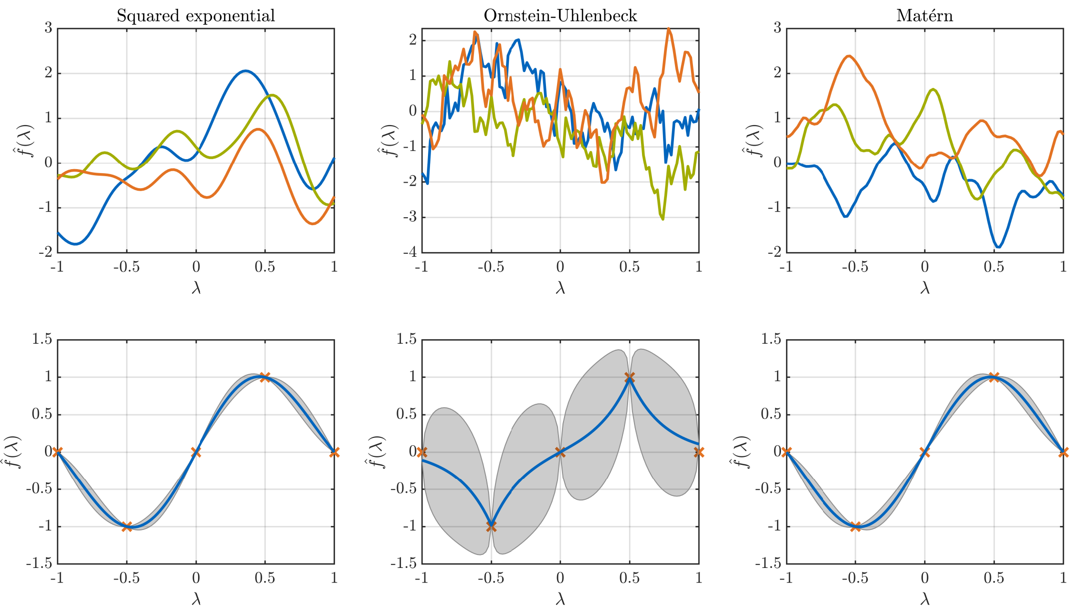


FIGURE 4.11: Top row: samples from a zero-mean Gaussian process using a squared exponential kernel ($\ell = 0.2$), an Ornstein-Uhlenbeck kernel ($\ell = 0.2$) and a Matérn kernel ($\ell = 0.2, \nu = 2.5$). Bottom row: GP regression on noise-free data from $f(\lambda) = \sin(2\pi\lambda)$ using the above kernel functions. The hyperparameters ℓ, ν were hand-picked.

with

$$\bar{\boldsymbol{C}}(\boldsymbol{\theta}_k^*) = \begin{pmatrix} \boldsymbol{C}(\boldsymbol{\theta}_k^*) & \boldsymbol{k} \\ \boldsymbol{k}^T & k(\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_q, \boldsymbol{\theta}_k^*) \end{pmatrix} \tag{4.143}$$

where $\boldsymbol{k}$ denotes a vector with components $k_n = k(\boldsymbol{\lambda}^{(n)}, \boldsymbol{\lambda}_q, \boldsymbol{\theta}_k^*)$. The predictive distribution $p(u_q|\mathcal{D}_\lambda, \hat{\boldsymbol{f}}, \boldsymbol{\lambda}_q)$ is a conditional Gaussian,

$$\begin{gathered} p(u_q|\mathcal{D}_\lambda, \hat{\boldsymbol{f}}, \boldsymbol{\lambda}_q) = \mathcal{N}(u_q|\mu_{\text{pred}}(\boldsymbol{\lambda}_q), \sigma^2_{\text{pred}}(\boldsymbol{\lambda}_q)), \\ \mu_{\text{pred}} = \boldsymbol{k}^T\boldsymbol{C}^{-1}\boldsymbol{u}, \qquad \sigma^2_{\text{pred}}(\boldsymbol{\lambda}_q) = k(\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_q) - \boldsymbol{k}^T\boldsymbol{C}^{-1}\boldsymbol{k}. \end{gathered} \tag{4.144}$$

A noise parameter can be added by adding a term $+\hat{\sigma}^2\delta(\boldsymbol{\lambda} - \boldsymbol{\lambda}')$ to the covariance function $k(\boldsymbol{\lambda}, \boldsymbol{\lambda}')$ and learn $\hat{\sigma}^2$ from the data according to Equation (4.140). Regression examples using different kernel functions with hand-picked hyperparameters $\boldsymbol{\theta}_k$ are shown in Figure 4.11.

It has to be noted that the gradient of the log (marginal) likelihood

$$\frac{\partial}{\partial\theta_{k,i}} \log \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}(\boldsymbol{\theta}_m), \boldsymbol{C}(\boldsymbol{\theta}_k)) = -\frac{1}{2}\text{Tr}\left(\boldsymbol{C}^{-1}\frac{\partial \boldsymbol{C}}{\partial\theta_{k,i}} + \frac{1}{2}\boldsymbol{u}^T\boldsymbol{C}^{-1}\frac{\partial \boldsymbol{C}}{\partial\theta_{k,i}}\boldsymbol{C}^{-1}\boldsymbol{u}\right) \tag{4.145}$$

involves inverting $\boldsymbol{C}$, which is a $\mathcal{O}(N^3)$ operation. Although several workarounds exist [233–235], Gaussian processes are typically inapplicable in big data problems where $N \gtrsim 10{,}000$.

Moreover, many popular and easy to handle kernel functions are stationary, $k(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = k(\boldsymbol{\lambda} - \boldsymbol{\lambda}')$. If, however, a GP regression is based on a similarity measure which

Distribution $\rho(\lambda)$	Support	gPC basis polynomials $\{\Psi_i\}_{i=0}^{\infty}$
Gaussian	$(-\infty, \infty)$	Hermite
Uniform	$[-1, 1]$	Legendre
Gamma	$[0, \infty)$	Laguerre
Beta	$[-1, 1]$	Jacobi

TABLE 4.1: Popular probability density metrics and the corresponding family of orthogonal polynomials [15].

itself is only based on relative distances $\boldsymbol{\lambda} - \boldsymbol{\lambda}'$ in the input space, it is by default prone to the curse of dimensionality. Thus, in high input dimensions, care has to be taken about the design of the kernel functions [236]. Other authors suggest built-in dimensionality reduction techniques [61, 237] or multi-fidelity information fusion [52, 56, 60] to overcome the problem.

4.5.3 Spectral methods

A popular class of surrogate models in the context of stochastic partial differential equations (SPDEs, see Section 3) are *(generalized) polynomial chaos* (gPC) *expansions* [14, 15, 17, 238], which replace the expensive, univariate forward model $u(\lambda)$ by an expansion in orthogonal polynomials $\Psi_p(\lambda)$,

$$u(\lambda) = \sum_{p=0}^{\infty} \hat{\theta}_p \Psi_p(\lambda) \approx \sum_{p=0}^{P} \hat{\theta}_p \Psi_p(\lambda), \tag{4.146}$$

with the orthonormality relation

$$\langle \Psi_i, \Psi_j \rangle_\rho = \int \Psi_i(\lambda) \Psi_j(\lambda) \rho(\lambda) d\lambda = \delta_{ij}, \tag{4.147}$$

i.e., $\{\Psi_p\}_{p=0}^{\infty}$ forms an orthonormal basis of the Hilbert space $\mathcal{H} =$
$= \operatorname{span}\left(\{\Psi_p(\lambda)\}_{p=0}^{\infty}\right)$ with the inner product

$$\langle v, w \rangle_\rho = \int v(\lambda) w(\lambda) \rho(\lambda) d\lambda \tag{4.148}$$

where $\rho(\lambda) \geq 0$ is a weight function usually assumed to be a valid probability density, i.e. $\int \rho(\lambda) d\lambda = 1$. Clearly, the particular form of orthogonal polynomials depends on the distribution that is assumed for $\rho(\lambda)$. The most common densities used in gPC and the corresponding polynomials are summarized in Table 4.1.

In practice, the infinite sum in Equation (4.146) needs to be truncated at a finite number P of terms,

$$\hat{f}(\lambda; \hat{\boldsymbol{\theta}}) = \sum_{p=0}^{P} \hat{\theta}_p \Psi_p(\lambda), \tag{4.149}$$

where typically only the P lowest order polynomials are kept. Besides finding good choices for the distribution/gPC basis pair from Table 4.1 and a suitable cutoff value

P, the central question is how to efficiently find the expansion coefficients $\hat{\theta}_p$.

The methods for finding $\hat{\theta}_p$ are commonly classified into *intrusive* and *non-intrusive* approaches (see e.g. [3, 4]). In intrusive gPC approaches, the expensive computer code $u(\lambda)$ that is to be replaced by the surrogate $\hat{f}$ needs to be modified, which is often impossible if commercial software for the model $u(\lambda)$ is used. On the other side, non-intrusive approaches only require sample evaluations $\left\{u(\lambda^{(n)})\right\}_{n=1}^{N}$ of the computer code $u(\lambda)$ s.t. it can be treated as a black box.

Non-intrusive methods

One way to get the coefficients $\hat{\theta}_k$ is by direct projection onto the basis function Ψ_k. From Equation (4.146), we get

$$\langle u, \Psi_k \rangle_\rho = \sum_{p=0}^{\infty} \hat{\theta}_p \left\langle \Psi_p, \Psi_k \right\rangle_\rho = \hat{\theta}_k. \tag{4.150}$$

This however requires explicit evaluation of the integral

$$\langle u, \Psi_k \rangle_\rho = \int u(\lambda)\Psi_k(\lambda)\rho(\lambda)d\lambda, \tag{4.151}$$

which is generally not given in closed form. Moreover, the forward model $u(\lambda)$ is assumed to be expensive to evaluate s.t. numerical approximations to Equation (4.151) like Monte Carlo or Gauss quadrature quickly become infeasible.

In *stochastic collocation* (SC) [24, 25, 239], a finite set of collocation points $\left\{\lambda^{(n)}\right\}_{n=1}^{N}$ are specified and the polynomial expansion coefficients are sought s.t. they exactly match the expensive forward model $u(\lambda)$ at the collocation points,

$$u(\lambda^{(n)}) = \hat{f}(\lambda^{(n)}; \hat{\boldsymbol{\theta}}) = \sum_{p=0}^{P} \hat{\theta}_p \Psi_p(\lambda^{(n)}). \tag{4.152}$$

Using Lagrange polynomials $\Psi_n(\lambda) = L_n(\lambda)$ with

$$L_n(\lambda) = \prod_{\substack{1 \le k \le N, \\ k \ne n}} \frac{\lambda - \lambda^{(k)}}{\lambda^{(n)} - \lambda^{(k)}}, \qquad L_n(\lambda^{(k)}) = \delta_{nk}, \tag{4.153}$$

the coefficients $\hat{\theta}_p$ would simply be the solution of the forward model u at the collocation point $\lambda^{(p)}$, $\hat{\theta}_p = u(\lambda^{(p)})$. For a general set of $P+1$ linearly independent basis polynomials $\left\{\Psi_p\right\}_{p=0}^{P}$, $N = P+1$ collocation points lead to the solution

$$\hat{\theta}_p = \sum_{n=1}^{N} \Phi_{pn}^{-1} u(\lambda^{(n)}), \tag{4.154}$$

where $\Phi_{pn} = \Psi_p(\lambda^{(n)})$ is called the Vandermonde matrix [16] in this context. It can be shown [16] that the interpolation error depends on the number of collocation points N and the stochastic dimension $d = \dim(\lambda)$ as

$$|u(\boldsymbol{\lambda}) - \hat{f}(\boldsymbol{\lambda})| \propto N^{-\alpha/d}, \tag{4.155}$$

i.e., to keep the error constant with increasing stochastic dimension d, the total number of collocation points N needs to grow exponentially – the method therefore strongly suffers from the curse of dimensionality. *Sparse grid stochastic collocation* [26, 27, 240] alleviates the issue, but only extends the range from ~ 5 to a few tens of stochastic input dimensions that are feasible with SC.

Another issue is that, as implied in Equation (4.154), SC uses as many collocation (or data) points N as there are free parameters $\hat{\theta}_p$ in the model which makes it is prone to overfitting. Another way to find the expansion coefficients $\hat{\theta}_p$ is therefore to perform a least-squares regression

$$\hat{\boldsymbol{\theta}}^* = \arg\min_{\hat{\boldsymbol{\theta}}} \sum_{n=1}^{N} \|u(\lambda^{(n)}) - \hat{f}(\lambda; \hat{\boldsymbol{\theta}})\|^2 \tag{4.156}$$

with $N \gtrsim 5(P+1)$ [241]. As this is nothing but a maximum likelihood approach to a Gaussian linear model with polynomial basis functions, the solution for $\hat{\boldsymbol{\theta}}^*$ is given in Equation (4.11). Also, L_1-regularization can be employed [242, 243] to generate sparse solutions, as discussed in Section 4.2.2.

Intrusive methods

Intrusive polynomial chaos or *Galerkin projection* is a surrogate modeling approach that is specific to the solution of SPDEs (see Section 3). For instance, consider a SPDE of the form

$$\begin{aligned} \mathcal{A}(\boldsymbol{x}, K(\boldsymbol{x},\lambda)u(\boldsymbol{x},\lambda)) = s(\boldsymbol{x}) \quad &\text{for} \quad \boldsymbol{x} \in \Omega, \\ \mathcal{B}(u) = 0 \quad &\text{for} \quad \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{4.157}$$

where $\mathcal{A}$ is a stochastic differential operator defined over the physical domain Ω, $\mathcal{B}$ specifies the boundary conditions on $\partial\Omega$ and $K(\boldsymbol{x},\lambda)$ is a random field (e.g., a PDE coefficient) depending on some uncertain parameter $\lambda \in \Lambda$. The standard approach is to use the same truncated polynomial chaos expansion for the solution field $u(\boldsymbol{x},\lambda)$ and the random field $K(\boldsymbol{x},\lambda)$[12],

$$u(\boldsymbol{x},\lambda) \approx \sum_{p=0}^{P} \hat{u}_p(\boldsymbol{x})\Psi_p(\lambda), \qquad K(\boldsymbol{x},\lambda) \approx \sum_{p=0}^{P} \hat{K}_p(\boldsymbol{x})\Psi_p(\lambda). \tag{4.158}$$

[12]Often, also the source term $s(\boldsymbol{x})$ and the boundary conditions $\mathcal{B}(u)$ are assumed to be random, i.e., $s(\boldsymbol{x}) = s(\boldsymbol{x},\lambda), \mathcal{B}(u) = \mathcal{B}(u,\lambda)$. For the sake of simplicity, we confine ourselves to deterministic right hand sides $s(\boldsymbol{x})$ and boundary conditions $\mathcal{B}(u)$ though.

The above approximations are plugged into the PDE Equation (4.157) and a Galerkin projection onto the component Ψ_k is performed

$$\begin{gathered}\left\langle \mathcal{A}\left(\boldsymbol{x}, \sum_p \hat{K}_p(\boldsymbol{x})\Psi_p(\lambda); \sum_p \hat{u}_p(\boldsymbol{x})\Psi_p(\lambda)\right), \Psi_k(\lambda)\right\rangle_\rho = \langle s(\boldsymbol{x}), \Psi_k\rangle_\rho, \\ \left\langle \mathcal{B}\left(\sum_p \hat{u}_p(\boldsymbol{x})\Psi_p(\lambda)\right), \Psi_k(\lambda)\right\rangle_\rho = 0\end{gathered} \tag{4.159}$$

to yield a system of $P+1$ coupled deterministic PDEs for the components $\hat{u}_p(\boldsymbol{x})$. It is noted that for the deterministic fields s and $\mathcal{B}$, $\langle s(\boldsymbol{x}), \Psi_0\rangle = s(\boldsymbol{x})$, $\langle \mathcal{B}(u), \Psi_0\rangle = \mathcal{B}(u)$ and $\langle s(\boldsymbol{x}), \Psi_k\rangle = 0$, $\langle \mathcal{B}(u), \Psi_k\rangle = 0$ for all $k \geq 1$. Efficient iterative solution schemes for the coupled system of elliptic PDEs have been proposed in [244, 245].

One major drawback of gPC methods is that, in their standard form, they are treated in a non-Bayesian way and therefore yield only point estimates instead of full predictive distributions. What is even worse is the scaling with the stochastic dimension $d_{\boldsymbol{\lambda}} = \dim(\boldsymbol{\lambda})$: the number of expansion components N_Ψ, i.e., the number of coupled PDEs in stochastic Galerkin as well as the minimum number of collocation points for the stochastic collocation approach grow quickly according to

$$N_\Psi = \binom{P + d_{\boldsymbol{\lambda}}}{P} = \frac{(P + d_{\boldsymbol{\lambda}})!}{P! d_{\boldsymbol{\lambda}}!}, \tag{4.160}$$

where P is the polynomial expansion order. This makes gPC methods particularly inefficient when $d_{\boldsymbol{\lambda}}$ is high.

4.5.4 The reduced basis method

Another class of approaches that can be viewed as surrogate models specifically designed for the solution of PDEs are *reduced basis* (RB) methods [29, 31, 32]. RB methods search for approximate solution fields $u_{RB}(\boldsymbol{x}, \lambda)$ in a reduced solution space $\mathbb{V}_{RB}$ spanned by a reduced basis[13]

$$\mathbb{V}_{RB} = \operatorname{span}\left(\psi_{RB,1}(\boldsymbol{x}), \ldots, \psi_{RB,L}(\boldsymbol{x})\right). \tag{4.161}$$

The RB functions $\psi_{RB,1}(\boldsymbol{x}), \ldots, \psi_{RB,L}(\boldsymbol{x})$ are extracted from a set of forward model evaluations, called *snapshots*, either by the greedy algorithm [33, 34] or by using a proper orthogonal decomposition (POD) [31, 32], which is described in the following.

[13]Note that the $\psi_{RB,k}$'s and ψ_k's discussed here have nothing to do with the Ψ_k's from the previous section.

It is assumed that the expensive full-order model (FOM) $u(\boldsymbol{x}, \boldsymbol{\lambda})$ is the solution to a fine scale finite element analysis,

$$u(\boldsymbol{x}, \boldsymbol{\lambda}) = \sum_{i=1}^{N_{dof}} \hat{u}_i(\boldsymbol{\lambda}) \psi_i(\boldsymbol{x}), \tag{4.162}$$

where $\hat{\boldsymbol{u}}(\boldsymbol{\lambda})$ is a collection of the *degrees of freedom* (dof's) called the solution vector and $\psi_i(\boldsymbol{x})$ are the fine scale finite element shape functions. In the RB literature, data accumulation and model training are called the *offline* stage, whereas performing predictions is called the *online* stage.

In the offline stage, data $\mathcal{D} = \left\{ \boldsymbol{\lambda}^{(n)}, \hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(n)}) \right\}_{n=1}^{N}$ are collected and assembled to a snapshot matrix $\boldsymbol{S} \in \mathbb{R}^{N_{dof} \times N}$,

$$\boldsymbol{S} = \left(\hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(1)}) \,, \; \dots \; , \hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(N)}) \right), \tag{4.163}$$

where generally the number of snapshots N is much smaller than the number of dof's, $N \ll N_{dof}$. The column space $\mathrm{col}(\boldsymbol{S})$ spans an N-dimensional subspace of $\mathbb{R}^{N_{dof}}$. The objective of the RB method using POD is to find the L-dimensional, orthogonal subspace $\mathrm{col}(\boldsymbol{V})$, with $L \leq N$, that best approximates the N-dimensional subspace $\mathrm{col}(\boldsymbol{S})$ in the sense that the square projection error is minimized,

$$\sum_{n=1}^{N} \| \hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(n)}) - \boldsymbol{V}\boldsymbol{V}^T \hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(n)}) \|^2 = \min_{\tilde{\boldsymbol{V}}} \sum_{n=1}^{N} \| \hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(n)}) - \tilde{\boldsymbol{V}}\tilde{\boldsymbol{V}}^T \hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(n)}) \|^2, \tag{4.164}$$

under the orthonormality condition $\tilde{\boldsymbol{V}}^T \tilde{\boldsymbol{V}} = \boldsymbol{I}_L$. According to the Schmidt-Eckardt-Young theorem [32, 246],

$$\sum_{n=1}^{N} \| \hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(n)}) - \boldsymbol{V}\boldsymbol{V}^T \hat{\boldsymbol{u}}(\boldsymbol{\lambda}^{(n)}) \|^2 = \sum_{n=L+1}^{N} s_n^2, \tag{4.165}$$

where

$$s_n^2 \boldsymbol{V}_{:n} = (\boldsymbol{S}\boldsymbol{S}^T) \boldsymbol{V}_{:n}, \tag{4.166}$$

i.e., to get the optimal subspace $\mathrm{col}(\boldsymbol{V})$, it is necessary to solve the above eigenvalue problem. Since the matrix $\boldsymbol{S}\boldsymbol{S}^T \in \mathbb{R}^{N_{dof} \times N_{dof}}$ is typically large, it is helpful to multiply Equation (4.166) by $\boldsymbol{S}^T$ from the left to get

$$s_n^2 (\boldsymbol{S}^T \boldsymbol{V}_{:n}) = (\boldsymbol{S}^T \boldsymbol{S})(\boldsymbol{S}^T \boldsymbol{V}_{:n}), \tag{4.167}$$

which is an eigenvalue problem to the usually much smaller matrix $\boldsymbol{S}^T \boldsymbol{S} \in \mathbb{R}^{N \times N}$. To obtain the eigenvectors $\boldsymbol{V}_{:n}$, multiply Equation (4.167) by $\boldsymbol{S}$ from the left to get

$$\boldsymbol{V}_{:n} = \frac{1}{s_n^2} \boldsymbol{S}(\boldsymbol{S}^T \boldsymbol{V}_{:n}). \tag{4.168}$$

The reduced dimension L may either be predefined or chosen as the minimum integer fulfilling

$$\frac{\sum_{n=L+1}^{N} s_n^2}{\sum_{n=1}^{N} s_n^2} < \delta_{\mathrm{POD}}^2, \tag{4.169}$$

where δ_{POD}^2 is a specified projection error.

As discussed in Section 3.1, a PDE of the form as, e.g., given in Equation (4.157) may be solved numerically by the finite element method leading to a system of linear[14] algebraic equations which may be written, in the original basis, as

$$\boldsymbol{A}(\boldsymbol{\lambda})\hat{\boldsymbol{u}} = \boldsymbol{F}(\boldsymbol{\lambda}), \tag{4.170}$$

where $\hat{\boldsymbol{u}}$ is the vector of dof's as defined by Equation (4.162), $\boldsymbol{A}(\boldsymbol{\lambda})$ is called the *stiffness matrix* and $\boldsymbol{F}(\boldsymbol{\lambda})$ the *right hand side* or *force vector*. Similar to a standard change of basis, the system given in Equation (4.170) can be projected onto the reduced space by

$$\boldsymbol{A}_{RB}(\boldsymbol{\lambda}) = \boldsymbol{V}^T\boldsymbol{K}(\boldsymbol{\lambda})\boldsymbol{V}, \qquad \boldsymbol{F}_{RB}(\boldsymbol{\lambda}) = \boldsymbol{V}^T\boldsymbol{F}(\boldsymbol{\lambda}) \tag{4.171}$$

which yields the reduced system

$$\boldsymbol{A}_{RB}(\boldsymbol{\lambda})\hat{\boldsymbol{u}}_{RB} = \boldsymbol{F}_{RB}(\boldsymbol{\lambda}) \tag{4.172}$$

of only L instead of N_{dof} equations. The approximate solution $\boldsymbol{f}(\boldsymbol{\lambda})$ can be reconstructed by solving the reduced system and back-projecting to the original space,

$$\hat{\boldsymbol{u}}_{RB}(\boldsymbol{\lambda}) = \boldsymbol{A}_{RB}^{-1}(\boldsymbol{\lambda})\boldsymbol{F}_{RB}(\boldsymbol{\lambda}), \qquad \hat{\boldsymbol{u}}(\boldsymbol{\lambda}) \approx \boldsymbol{f}(\boldsymbol{\lambda}) = \boldsymbol{V}\hat{\boldsymbol{u}}_{RB}(\boldsymbol{\lambda}). \tag{4.173}$$

As the reduced system contains only L instead of N_{dof} linear equations, it scales as (worst case) $\mathcal{O}(L^3) \ll \mathcal{O}(N_{dof}^3)$ only. However, the projection $\boldsymbol{V}^T\boldsymbol{A}(\boldsymbol{\lambda})\boldsymbol{V}$ scales as $\mathcal{O}(N_{dof}^2)$ and needs to be evaluated online, i.e., for every new input $\boldsymbol{\lambda}$.

To overcome this restraint, many FEM systems allow to apply the affine decomposition

$$\boldsymbol{A}_{RB}(\boldsymbol{\lambda}) = \sum_{i=1}^{N_A} \alpha_i(\boldsymbol{\lambda})\boldsymbol{A}_{RB}^{(i)}, \qquad \boldsymbol{F}_{RB}(\boldsymbol{\lambda}) = \sum_{i=1}^{N_F} \beta_i(\boldsymbol{\lambda})\boldsymbol{F}_{RB}^{(i)}, \tag{4.174}$$

where $\alpha_i(\boldsymbol{\lambda}), \beta_i(\boldsymbol{\lambda})$ are some coefficients and the $\boldsymbol{A}_{RB}^{(i)}$'s, $\boldsymbol{F}_{RB}^{(i)}$'s are independent of the input parameters $\boldsymbol{\lambda}$ and can therefore be precomputed offline. This way, the numerical cost of the online stage is independent of N_{dof}. Details can be found in [31, 32]. Although rigorous error bounds exist [247], a fully probabilistic treatment of the basis reduction, PDE solution and reconstruction is still missing.

[14]FEM discretization may lead to a nonlinear system of equations. We confine ourselves to a discussion of the RB method in linear problems only.

Chapter 5

A physics-aware machine learning framework for high-dimensional systems in the Small Data regime

This chapter is based on the findings published in

C. Grigo, P.-S. Koutsourelakis:

"A physics-aware, probabilistic machine learning framework for coarse-graining high-dimensional systems in the Small Data regime",
Elsevier Journal of Computational Physics 2019, Volume 397, 108842

C. Grigo, P.-S. Koutsourelakis:

"Bayesian Model and Dimension Reduction for Uncertainty Propagation: Applications in Random Media",
SIAM/ASA Journal on Uncertainty Quantification 2019 7:1, 292-323

C. Grigo, P.-S. Koutsourelakis:

"Probabilistic reduced-order modeling for stochastic partial differential equations",
Eccomas Proceedia UNCECOMP (2017) 111-129

As discussed in Section 4.5, surrogate modeling of complex computer codes using standard machine learning algorithms such as artificial neural networks (Section 4.5.1) or Gaussian process regression (Section 4.5.2) is strongly aggravated in high-dimensional input/output settings due to the curse of dimensionality. Moreover, because of the large numerical cost, often only a few hundreds or even less forward model runs can practically be performed for data acquisition. Even rather SPDE-specific approaches such as (intrusive) gPC (Section 4.5.3) or the reduced basis method (Section 4.5.4) typically struggle with the "high dimension, small data" setting.

In such situations, it is of primary interest to design surrogate models that incorporate as much as possible a priori information of the underlying physical process directly into their model structure. Such methods have become very popular only recently under the catchwords "physics-informed", "physics-constrained" or "physics-aware" machine learning, primarily constrained to ANN [81–85] or GP models [62, 248], where the basic principle is to enrich the training process by minimization of the residuals to the governing equations, potentially making classic forward model evaluation data unnecessary at the expense of transferring much of the numerical cost to the repeated residual evaluation [73, 86].

However, machine learning models that integrate physical principles directly by model construction, i.e., without any additional cost during training or prediction stages, are still underexplored. The approach presented in this chapter takes a step in this direction by outlining a model that, as a central unit, contains a stencil solver based on simplified constitutive physics as well as lower spatio-temporal resolution.

For the paradigmatic problems of Stokes and Darcy flow through random heterogeneous media discussed in this work, the model is capable to yield accurate probabilistic predictions in a high-dimensional ($\gtrsim 10^4$), small data ($N \lesssim 100$) regime even under explorative conditions far away from the training data. To achieve this, it is indispensable that (a) the machine learning surrogate is able to extract only the properties from the porous microstructure that are most relevant for the reconstruction of the fine-grained model (FGM) output (i.e., pressure and velocity fields $P, \boldsymbol{V}$, see Chapter 2), and (b) only allows for predictive responses that are in accordance with basic physical principles. Whilst (a) is essentially equivalent to a dimension reduction of the high-dimensional description of the random microstructures, (b) excludes a wide variety of possible predictions that are implausible with regard to the governing equations of the problem.

5.1 Notation and preliminary remarks

The objective is to design a surrogate model for both Darcy and Stokes flow through random media, see Chapters 2 and 3 for the governing equations and numerical solutions. We denote by $\boldsymbol{u}_f$ the quantity of interest that is to be predicted by the surrogate. Typically, $\boldsymbol{u}_f$ corresponds to the pressure response of the FGM evaluated on a regular fine scale grid $G^{(f)}$, i.e., $u_{f,i} = P(\boldsymbol{x}_i), \boldsymbol{x}_i \in G^{(f)}$.

For the moment, we assume that boundary conditions are fixed so that the only varying input to the FGM is the porous microstructure described by the high-dimensional input vector $\boldsymbol{\lambda}_f$. The random microstructure $\boldsymbol{\lambda}_f$ can either be described as a digitized image of pixels (or voxels) being either black for solid or white for fluid spaces (as depicted in Figure 2.4), in which case $\boldsymbol{\lambda}_f$ is given by a binary list of pixel values. Although we only consider binary materials here, it is noted that for Darcy flow, the permeability field can vary continuously, resulting in a non-binary vector $\boldsymbol{\lambda}_f$. Or,

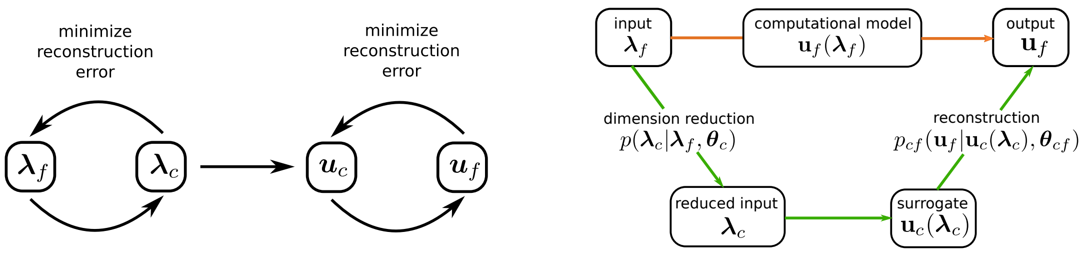

minimize reconstruction error
minimize reconstruction error
λ_f
λ_c
u_c
u_f
input $\boldsymbol{\lambda}_f$
computational model $\mathbf{u}_f(\boldsymbol{\lambda}_f)$
output $\mathbf{u}_f$
dimension reduction $p(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c)$
reconstruction $p_{cf}(\mathbf{u}_f|\mathbf{u}_c(\boldsymbol{\lambda}_c), \boldsymbol{\theta}_{cf})$
reduced input $\boldsymbol{\lambda}_c$
surrogate $\mathbf{u}_c(\boldsymbol{\lambda}_c)$

FIGURE 5.1: Left: Schematic representation of the traditional, separated dimension-reduction/surrogate modeling approach where both steps are disconnected and the reduced representation $\boldsymbol{\lambda}_c$ is searched to minimize the reconstruction error to $\boldsymbol{\lambda}_f$ and then used as the input to a surrogate denoted by $\boldsymbol{u}_c$. Right: Connected approach – the reduced representation $\boldsymbol{\lambda}_c$ is searched such that it is maximally informative for the reconstruction of the response $\boldsymbol{u}_f$, not the input $\boldsymbol{\lambda}_f$.

considering microstructures with random non-overlapping spherical exclusions as, e.g., shown in Figure 2.5, the microstructure can be fully described by a list of radii and center coordinates of the corresponding circular exclusions. In that case, we have that $\dim(\boldsymbol{\lambda}_f) = 3N_{\text{excl}}$, where N_{excl} is the number of spherical exclusions of the porous domain. If N_{excl} is assumed to be random, it is obvious that $\dim(\boldsymbol{\lambda}_f)$ will not be constant which, as will become clear later, is by no means necessary. In any case, $\boldsymbol{\lambda}_f$ may always be transformed to a digitized image with constant $\dim(\boldsymbol{\lambda}_f)$ if desired.

Given this notation, the FGM is written down as the mapping

$$\boldsymbol{u}_f(\boldsymbol{\lambda}_f) : \boldsymbol{\lambda}_f \mapsto \boldsymbol{u}_f, \tag{5.1}$$

and the dataset $\mathcal{D}$ the surrogate will be trained on is given by N FGM evaluations, $\mathcal{D} = \left\{\boldsymbol{\lambda}_f^{(n)}, \boldsymbol{u}_f(\boldsymbol{\lambda}_f^{(n)})\right\}_{n=1}^{N}$, where $\boldsymbol{\lambda}_f^{(n)} \sim p_{\boldsymbol{\lambda}_f}(\boldsymbol{\lambda}_f)$.

Due to the high dimensionality of $\boldsymbol{\lambda}_f$ and in order for the surrogate model to generalize well, it is imperative that a dimension reduction step is contained in the model to find a reduced representation $\boldsymbol{\lambda}_c$ of $\boldsymbol{\lambda}_f$ with $\dim(\boldsymbol{\lambda}_c) \ll \dim(\boldsymbol{\lambda}_f)$. A common approach is to perform the dimension reduction using only information from the model inputs, see the left part of Figure 5.1, with, e.g., a principal component analysis. This is suboptimal since it discounts the actual purpose of the dimension reduction, which is the extraction of all relevant information in order to reconstruct the FGM *output* $\boldsymbol{u}_f$, not the *input* $\boldsymbol{\lambda}_f$ itself. The approach we follow here is depicted in the right part of Figure 5.1: dimension reduction and surrogate modeling need to be connected and trained in a single model in order to find the *joint optimum* of a reduced representation $\boldsymbol{\lambda}_c$ and a reconstruction to $\boldsymbol{u}_f$.

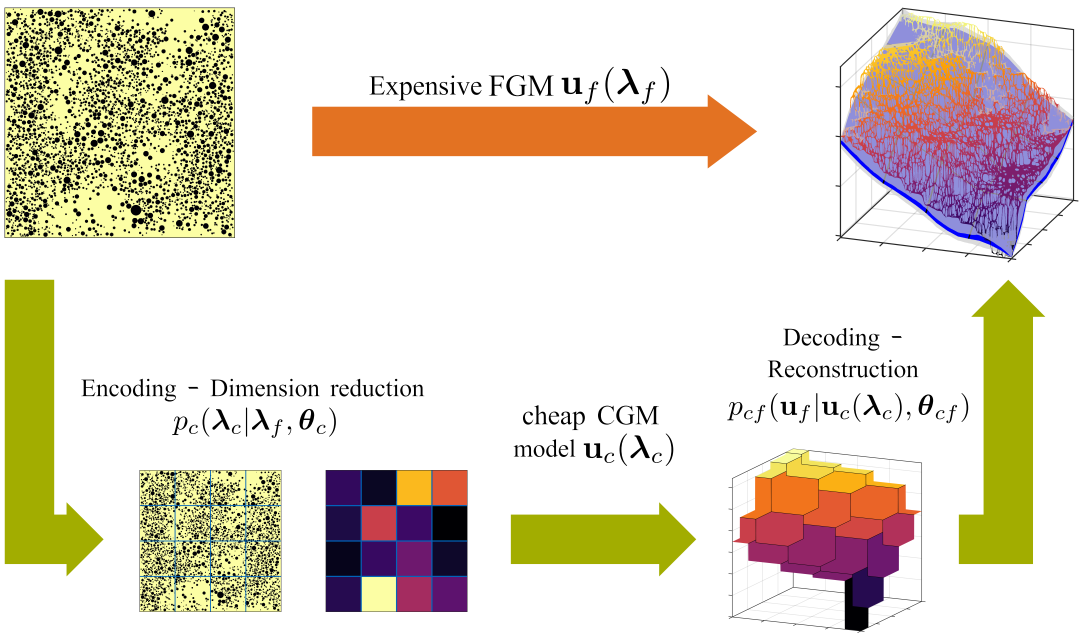

Expensive FGM $\mathbf{u}_f(\boldsymbol{\lambda}_f)$
Encoding - Dimension reduction
$p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f,\boldsymbol{\theta}_c)$
cheap CGM
model $\mathbf{u}_c(\boldsymbol{\lambda}_c)$
Decoding -
Reconstruction
$p_{cf}(\mathbf{u}_f|\mathbf{u}_c(\boldsymbol{\lambda}_c),\boldsymbol{\theta}_{cf})$

FIGURE 5.2: Graphical representation of the proposed three-component model. Instead of solving the expensive FGM $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$ directly (orange arrow), we first find an effective, low-dimensional representation $\boldsymbol{\lambda}_f$ of the high-dimensional microstructure $\boldsymbol{\lambda}_f$ via the conditional distribution $p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f,\boldsymbol{\theta}_c)$. Next, a coarse-grained model (CGM) $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ is solved using much coarser discretizations and possibly simplified governing equations. Finally, the CGM output $\boldsymbol{u}_c$ is reconstructed via $p_{cf}(\boldsymbol{u}_f|\boldsymbol{u}_c,\boldsymbol{\theta}_{cf})$ to yield a predictive distribution over the FGM response $\boldsymbol{u}_f$. Picture taken from [87].

5.2 Model architecture

Due to the aforementioned desiderata, we propose a fully probabilistic three-component machine learning model that is constituted of the following key elements [87, 103, 104]:

- **Encoding/dimension reduction:** a probabilistic mapping from the high-dimensional stochastic input $\boldsymbol{\lambda}_f$ to a much lower-dimensional, encoded representation $\boldsymbol{\lambda}_c$. This mapping shall retain as much as possible information from $\boldsymbol{\lambda}_f$ for the reconstruction of the true response $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$. It is mediated by the conditional density $p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f,\boldsymbol{\theta}_c)$ controlled by the parameters $\boldsymbol{\theta}_c$;

- **Coarse-grained solver:** A coarse-grained model (CGM) operating on larger length scales and possibly simplified constitutive equations is the central component of the model. For a general, probabilistic CGMs, we denote the associated input/output mapping by $p_{\mathrm{CGM}}(\boldsymbol{u}_c|\boldsymbol{\lambda}_c)$;

- **Decoding/reconstruction:** A probabilistic mapping from the CGM output $\boldsymbol{u}_c$ to the true FGM response $\boldsymbol{u}_f$ mediated by a probabilistic mapping $p_{cf}(\boldsymbol{u}_f|\boldsymbol{u}_c,\boldsymbol{\theta}_{cf})$ controlled by the parameters $\boldsymbol{\theta}_{cf}$. The decoder mapping p_{cf} should take into account the smoothness/spatial character of the FGM response.

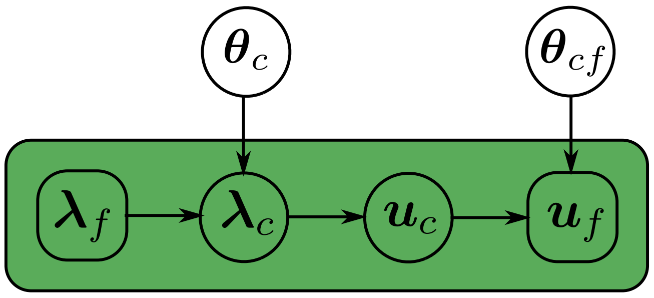


FIGURE 5.3: Graphical representation of the three-component Bayesian network implied by $p(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_{cf}, \boldsymbol{\theta}_c)$ in Equation (5.2). The internal vertices $\boldsymbol{\lambda}_c$, $\boldsymbol{u}_c$ are latent variables. Picture taken from [104].

A graphical illustration of the suggested model can be viewed in Figure 5.2.

The connection of the above densities gives

$$p(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c, \boldsymbol{\theta}_{cf}) = \int p_{cf}(\boldsymbol{u}_f|\boldsymbol{u}_c, \boldsymbol{\theta}_{cf}) p_{\text{CGM}}(\boldsymbol{u}_c|\boldsymbol{\lambda}_c) p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c) d\boldsymbol{u}_c d\boldsymbol{\lambda}_c. \tag{5.2}$$

Although a model based on a probabilistic CGM $p_{\text{CGM}}(\boldsymbol{u}_c|\boldsymbol{\lambda}_c)$ is conceivable, we assume a deterministic CGM $\boldsymbol{u}_c = \boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ for the rest of this work such that $p_{\text{CGM}}(\boldsymbol{u}_c|\boldsymbol{\lambda}_c) = \delta(\boldsymbol{u}_c - \boldsymbol{u}_c(\boldsymbol{\lambda}_c))$ and the above equation simplifies to

$$p(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c, \boldsymbol{\theta}_{cf}) = \int p_{cf}(\boldsymbol{u}_f|\boldsymbol{u}_c(\boldsymbol{\lambda}_c), \boldsymbol{\theta}_{cf}) p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c) d\boldsymbol{\lambda}_c. \tag{5.3}$$

This equation can be viewed as the *likelihood* of a single FGM data pair $\{\boldsymbol{\lambda}_f, \boldsymbol{u}_f\}$ and is thus the crucial ingredient of the proposed model. The probabilistic graphical model [249] defined by Equation (5.2) is depicted schematically in Figure 5.3.

It is noted that the three components are modular in the sense that they can easily be adapted to various kinds of FGM data. For example, similar model architectures have been used in other studies, e.g., for time dependent problems [250] or atomistic modeling of materials [251, 252]. Moreover, the specific probabilistic models (e.g., ANNs, GPs, linear models...) for p_c, p_{cf} are not determined by the above structure providing enough freedom to adapt the model to the FGM data under investigation.

The latent variables $\boldsymbol{\lambda}_c$ can be seen as a compressed, encoded version of the FGM input $\boldsymbol{\lambda}_f$ which are, at low computational cost, transformed to $\boldsymbol{u}_c$ by a simplified physics model $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ which is finally decoded to the FGM output $\boldsymbol{u}_f$ by the probabilistic coarse-to-fine mapping p_{cf}. It is again emphasized that for accurate predictions of $p(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c, \boldsymbol{\theta}_{cf})$, it is unimportant if $\boldsymbol{\lambda}_c$ is a high-fidelity encoder in the sense of low reconstruction error of the FGM *input* $\boldsymbol{\lambda}_f$. Instead, $\boldsymbol{\lambda}_c$ must be a high-quality encoder of the FGM *output* $\boldsymbol{u}_f$. In that sense, $\boldsymbol{\lambda}_c$ can be very different from a pure (linear or nonlinear) dimension reduction of $\boldsymbol{\lambda}_f$. A detailed description of the encoding density $p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c)$ as used in the numerical examples of Chapter 6 is given in Section 5.2.1.

It is important to note that the coarse-graining procedure $\boldsymbol{\lambda}_f \mapsto \boldsymbol{\lambda}_c$ comes with an unavoidable amount of information loss (unless redundancies are present in $\boldsymbol{\lambda}_f$) since for $\dim(\boldsymbol{\lambda}_c) < \dim(\boldsymbol{\lambda}_f)$ the mutual information $I(\boldsymbol{\lambda}_c, \boldsymbol{\lambda}_f)$ is bounded from above, $I(\boldsymbol{\lambda}_c, \boldsymbol{\lambda}_f) \leq I_0$ [253]. This means that, even in the limit of infinite training data $N \to \infty$, there will still be a residual amount of predictive uncertainty which is to be captured in the above densities. As for any probabilistic model of finite complexity, one may therefore discern between uncertainties due to (a) finite data and (b) model inadequacy.

The decoding step $\boldsymbol{u}_c \mapsto \boldsymbol{u}_f$, mediated by the decoding density p_{cf}, can be seen as a generative process for the FGM output. As $\dim(\boldsymbol{u}_c) \ll \dim(\boldsymbol{u}_f)$, the proposed framework will tend to smooth out predictions of $\boldsymbol{u}_f$ because of its incapability to fully resolve responses of higher dimension as $\boldsymbol{u}_c$. Because of the spatial structure $u_{f,i} = P(\boldsymbol{x}_i)$, $\boldsymbol{x}_i \in G^{(f)}$, it appears most natural to adopt a model for p_{cf} that plays the role of an interpolant, i.e., the CGM response components $u_{c,j} = P_c(\boldsymbol{x}_j)$, $\boldsymbol{x}_j \in G^{(c)}$, with $G^{(c)}$ a coarse grid corresponding to the discretization of $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$, should be more important the smaller the distance $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ between fine- and coarse-scale evaluation locations $\boldsymbol{x}_i, \boldsymbol{x}_j$ is. A detailed presentation of the decoding density p_{cf} as applied in Chapter 6 is given in Section 5.2.2.

The central component of the model is the CGM solver $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ which also determines the physical interpretation of $\boldsymbol{\lambda}_c$ and its relation to $\boldsymbol{\lambda}_f$. Apart from the constraint that the CGM $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ needs to be much cheaper to evaluate than the FGM $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$, several choices are possible. The most direct approach is to use a CGM that corresponds to a numerical solver to the same PDE as in the FGM, but with a much coarser spatio-temporal discretization. Other possibilities are given by different physics models, i.e., different governing equations for CGM and FGM, or fully stochastic models $p_{\mathrm{CGM}}(\boldsymbol{u}_c|\boldsymbol{\lambda}_c)$ based on, e.g., the residuals of an iterative FEM solver. In this work, we confine ourselves to deterministic CGMs $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ based on coarser discretizations and possibly different governing equations, namely Stokes and Darcy flow, see Chapter 2. The exact specifics are given in the corresponding experiments of Chapter 6.

5.2.1 The encoder distribution p_c

The encoder distribution $p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c)$ constitutes the probabilistic mapping from the high-dimensional, fine-scale description of the microstructure $\boldsymbol{\lambda}_f$ to the much lower-dimensional, effective representation $\boldsymbol{\lambda}_c$ and can assume many different forms. It should be capable to extract as good as possible the microstructural features from $\boldsymbol{\lambda}_f$ that are most predictive for the FGM output. One possible choice is to make use of the expressivity of artificial neural networks (ANNs) or, given the spatial nature of the random input $\boldsymbol{\lambda}_f$, to use convolutional neural networks (CNNs) [65, 67, 254].

However, given the rich literature on random heterogeneous media (see, e.g., [1] and references therein), using an ANN would be tantamount to a waste of a sizable amount of given a priori information about microstructure/property connections: for many material properties such as electrical or thermal conductivity, diffusion coefficients or elastic moduli, there exists a wide range of identified relevant topological features [255–257], effective property approximation formulas [258–260], or rigorous bounds [261] that could/should be hard-coded to the model to enrich it with the available amount of background knowledge.

For the encoding distribution p_c, we thus suggest a Gaussian linear model of physically motivated feature functions $\varphi_{jm}(\boldsymbol{\lambda}_f)$ presented in detail in Section 5.2.1 which formally takes the form

$$\lambda_{c,m} = \sum_{j=1}^{N^{(m)}_{\text{features}}} \tilde{\theta}_{c,jm}\varphi_{jm}(\boldsymbol{\lambda}_f) + \tau_{c,m}^{-1/2} Z_m \qquad Z_m \sim \mathcal{N}(0,1), \tag{5.4}$$

where $\boldsymbol{\theta}_c = \left\{\tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c^{-1}\right\}$ are modeling parameters to be learned from the data[1]. Easily graspable feature functions are, for instance, solid/fluid-phase volume fractions or expected 2-point correlations of the given microstructure $\boldsymbol{\lambda}_f$. In principle, different sets of feature functions $\boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f) = \begin{pmatrix} \varphi_{1m}(\boldsymbol{\lambda}_f) \\ \vdots \\ \varphi_{N^{(m)}_{\text{features}m}} \end{pmatrix}$ can be applied for the different latent space components m. Moreover, the coefficients $\tilde{\theta}_{c,jm}$ are generally different for different components m, i.e., $\tilde{\theta}_{c,jm} \neq \tilde{\theta}_{c,jm'}$ for $m \neq m'$, although a tied version $\tilde{\theta}_{c,jm} = \tilde{\theta}_{c,jm'}$ may be a reasonable strategy if model complexity needs to be reduced. The linear combination of feature functions is augmented by uncorrelated Gaussian white noise of variance $\tau_{c,m}^{-1}$ for the component m such that the distribution p_c is

$$p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c^{-1}) = \prod_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \mathcal{N}(\lambda_{c,m}|\tilde{\boldsymbol{\theta}}_{c,m}^T \boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f), \tau_{c,m}^{-1}) \tag{5.5}$$

with the feature function vector $\boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f)$ for component $\lambda_{c,m}$ and $\tilde{\boldsymbol{\theta}}_{c,m}$ the corresponding coefficients.

One may argue that a model of the type given in Equation (5.5) with hand-crafted feature functions $\varphi_{jm}(\boldsymbol{\lambda}_f)$ requires a lot of trial-and-error experiments/experience from the analyst to find a set of feature functions $\{\boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f)\}_{m=1}^{\dim(\boldsymbol{\lambda}_c)}$ that leads to accurate predictions of the overall surrogate $p(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_{cf}, \boldsymbol{\theta}_c)$ – as opposed to generic models like ANNs or GPs which do not require any a priori model selection. As will become clear in Section 4.2.2, this is invalid because sparsity enforcing prior models will be applied that automatically detect the most relevant feature functions out of a large library of features while pruning irrelevant ones by setting the corresponding coefficients $\tilde{\theta}_{c,jm} = 0$. The size of the feature library will only affect the

[1]We use the notation $\boldsymbol{\Sigma}_c = \operatorname{diag}(\boldsymbol{\tau}_c^{-1})$ with $\boldsymbol{\tau}_c^{-1} = \begin{pmatrix} \tau_{c,1}^{-1} \\ \vdots \\ \tau_{c,M}^{-1} \end{pmatrix}$, $M = \dim(\boldsymbol{\lambda}_c)$ wherever it is more convenient

training time, but is irrelevant for prediction because the features corresponding to $\tilde{\theta}_{c,jm} = 0$ not necessarily need to be considered. On the other side, also the network architecture of an ANN (number, size and type of layers, regularization, etc.) or the covariance function of a GP need to be pre-specified and hard-coded by the user, which is equivalent to model selection by feature engineering.

From microstructure encoding to effective material properties

As discussed above, the low-dimensional, encoded representation $\boldsymbol{\lambda}_c$ of a microstructure $\boldsymbol{\lambda}_f$ serves as the input to a coarse-grained, effective solver $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ which is based on coarser spatio-temporal discretizations and possibly simplified governing equations. Therefore, there needs to be a valid translation of the encoded representation $\boldsymbol{\lambda}_c$ to effective material parameters $\boldsymbol{K}$ the coarse-grained solver can work with.

A key issue of the $\boldsymbol{\lambda}_c$ to $\boldsymbol{K}$ translation is that material parameters $\boldsymbol{K}$ such as thermal or electric conductivity, magnetic/fluid permeability, dielectric constants or elastic moduli often need to comply with constraints of the form $K_m > 0$ or $K_{lo} < K_m < K_{hi}$ for scalar quantities K_m or $\boldsymbol{K}_m = pos.\ def.$ for second order tensor material properties. This can be achieved using a link functions $\chi(\lambda_{c,m})$ of the form

$$\begin{aligned}
K_m &= \chi(\lambda_{c,m}) = e^{\lambda_{c,m}} && \text{for} \quad K_m > 0, \\
K_m &= \chi(\lambda_{c,m}) = K_{lo} + (K_{hi} - K_{lo}) \frac{e^{\lambda_{c,m}}}{1 + e^{\lambda_{c,m}}} && \text{for} \quad K_{lo} < K_m < K_{hi}, \\
\boldsymbol{K}_m &= \chi(\lambda^{11}_{c,m}, \lambda^{12}_{c,m}, \lambda^{22}_{c,m}) = \begin{pmatrix} \lambda^{11}_{c,m} & 0 \\ \lambda^{12}_{c,m} & \lambda^{22}_{c,m} \end{pmatrix} \begin{pmatrix} \lambda^{11}_{c,m} & \lambda^{12}_{c,m} \\ 0 & \lambda^{22}_{c,m} \end{pmatrix} && \text{for} \quad \boldsymbol{K}_m = pos.def.,
\end{aligned} \tag{5.6}$$

where the last one corresponds to a Cholesky factorization $\boldsymbol{K} = \boldsymbol{L}_m \boldsymbol{L}_m^T$. For $\boldsymbol{K}_m, \boldsymbol{L}_m \in \mathbb{R}^{d \times d}$, this would correspond to a $\frac{1}{2}(d+1)d$ times higher latent space dimension $\dim(\boldsymbol{\lambda}_c)$ compared to the case of scalar/isotropic quantities K_m.

In the experiments carried out in Chapter 6, $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ corresponds to a finite element solver for Darcy flow as given in Equation (2.6), where the effective material property modeled by $\boldsymbol{\lambda}_c$ is the permeability field $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c)$. The problem domain Ω is subdivided into $N_{\text{cells},c} = \dim(\boldsymbol{\lambda}_c)$ of non-overlapping subregions $\{\Omega_m\}_{m=1}^{\dim(\boldsymbol{\lambda}_c)}$ such that $\bigcup_{m=1}^{N_{\text{cells, c}}} \Omega_m = \Omega$ and $\Omega_m \cap \Omega_{m'} = \emptyset$ for $m \neq m'$. In every subregion $\Omega_m \subset \Omega$, the permeability tensor is modeled to be constant, i.e.,

$$\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c) = \sum_{m=1}^{N_{\text{cells, c}}} = \mathbb{1}_{\Omega_m}(\boldsymbol{x}) \boldsymbol{K}_m(\boldsymbol{\lambda}_c), \tag{5.7}$$

where $\mathbb{1}_{\Omega_m}(\boldsymbol{x})$ denotes the indicator function for $\boldsymbol{x}$ to be in subregion Ω_m. To obtain physically meaningful solutions to Darcy flow Equation (2.6), it needs to be ensured that $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c) = pos.def.$ which is accomplished by the assumption of an isotropic material such that $\boldsymbol{K}_m(\boldsymbol{\lambda}_c)$ can be written as

$$\boldsymbol{K}_m(\boldsymbol{\lambda}_c) = \boldsymbol{K}_m(\lambda_{c,m}) = e^{\lambda_{c,m}}\boldsymbol{I} = K_m\boldsymbol{I}, \tag{5.8}$$

having the convenient property that one component $\lambda_{c,m}$ controls the effective permeability tensor $\boldsymbol{K}_m$ for every subregion Ω_m. In Chapter 6, the subregions Ω_m are modeled as squares. It is noted that the effective permeability field discretization implied by Equation (5.7) not necessarily needs to coincide with the finite element discretization used to solve the CGM problem. Obviously, many other choices of representations of $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c)$ are possible, e.g., an expansion in appropriate basis functions, as long as they ensure positive definiteness of $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c)$.

Microstructural feature functions

It is clear that the expressivity of the encoder distribution p_c hinges on the library of feature functions $\{\boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f)\}_{m=1}^{\dim(\lambda_c)}$. A reasonable strategy is thus to use as many feature functions as affordable (typically the limiting factor is increasing training time) that can be found in the literature.

In the numerical experiments of Chapter 6, we use topological descriptors such as n-point correlation functions [257], surface area and pore size density [255], chord-length densities [262] or lineal path functions [256]. As we are investigating flow problems through random porous media, another class of feature functions is given by well-known quantities from fluid dynamics such as the Poiseuille law [94] or the Kozeny-Carman equation [95–97]. Also, effective medium approximations from other physics, such as Maxwell's approximation [259], Archie's law [98, 263] from classical electrodynamics or the Bruggeman formula [260, 264, 265] may be applied. Other features are based on image recognition tools [93] or auto-encoder latent space representations [99, 226]. As they play a pivotal role in the surrogate modeling framework presented in this work, we dedicate ourselves to a short presentation of the most important feature functions that were used during this study and partially found their way into the examples of Chapter 6.

Unless otherwise noted, the feature functions $\varphi_{jm}(\boldsymbol{\lambda}_f)$ are evaluated only on the subregion Ω_m of the corresponding effective material property field discretization, i.e., $\varphi_{jm}(\boldsymbol{\lambda}_f) = \varphi_{jm}(\boldsymbol{\lambda}_{f,m})$ where $\boldsymbol{\lambda}_{f,m}$ is the part of $\boldsymbol{\lambda}_f$ that determines the microstructure in subregion Ω_m.

New feature functions can be constructed straightforwardly by composition with linearly independent[2] functions such as $\log(\varphi_{jm}(\boldsymbol{\lambda}_f))$, $\exp(\varphi_{jm}(\boldsymbol{\lambda}_f))$, $\sqrt{\varphi_{jm}(\boldsymbol{\lambda}_f)}$, $(\varphi_{jm}(\boldsymbol{\lambda}_f))^2, \dots$.

[2]If there are linear dependent features in a linear model, the maximum likelihood estimate for $\tilde{\boldsymbol{\theta}}_c$ would be non-unique. Obviously, this degeneracy is lifted by introduction of a prior on the parameters $\tilde{\boldsymbol{\theta}}_c$. Nevertheless, linearly dependent feature functions should be avoided because they contain redundant information.

Constant term It is advisable to include a feature function $\varphi_{jm}(\boldsymbol{\lambda}_f)$ that models a constant offset which is independent of the underlying microstructure $\boldsymbol{\lambda}_f$, i.e.,

$$\varphi_{jm}(\boldsymbol{\lambda}_f) = 1. \tag{5.9}$$

Using such a feature function exclusively would assign the same effective material property in any subregion Ω_m of any training sample n.

Volume fraction A simple, but quite expressive class of feature functions φ is based on the volume fractions $|\Omega_{s,m}|/|\Omega_m|$, $|\Omega_{f,m}|/|\Omega_m| = 1 - |\Omega_{s,m}|/|\Omega_m|$ of solid (or weakly permeable) and fluid (or highly permeable) phases in subregion m.

Effective medium approximations For many problems in random heterogeneous media, there exist approximation formulas for effective material properties based on the topology of the microstructure $\boldsymbol{\lambda}_f$.

One example is the Maxwell-Garnett approximation (MGA) [259] for the effective electrical conductivity λ_{eff} in a two-phase random medium with spherical inclusions of conductivity λ_{inc} in a matrix of conductivity λ_{mat}. In $2D$, the approximation formula is given by [1]

$$\lambda_{\text{eff}} = \lambda_{\text{mat}} \frac{\lambda_{\text{inc}} + \lambda_{\text{mat}} + \frac{|\Omega_{\text{inc}}|}{|\Omega|}(\lambda_{\text{inc}} - \lambda_{\text{mat}})}{\lambda_{\text{inc}} + \lambda_{\text{mat}} - \frac{|\Omega_{\text{inc}}|}{|\Omega|}(\lambda_{\text{inc}} - \lambda_{\text{mat}})}, \tag{5.10}$$

where $\frac{|\Omega_{\text{inc}}|}{|\Omega|}$ is the volume fraction of the inclusion phase. This can be directly used as a feature function of the form $\varphi_{jm}(\boldsymbol{\lambda}_f) = \lambda_{\text{eff}}$ or used in a composition with any other nonlinear function[3].

Another example is the self-consistent approximation (SCA) or Bruggeman formula [260]

$$\begin{aligned} \lambda_{\text{eff}} &= \frac{1}{2}\left(\alpha + \sqrt{\alpha^2 + 4\lambda_{\text{mat}}\lambda_{\text{inc}}}\right) \\ \alpha &= \lambda_{\text{mat}}\left(2\frac{|\Omega_{\text{mat}}|}{|\Omega|} - 1\right) + \lambda_{\text{inc}}\left(2\frac{|\Omega_{\text{inc}}|}{|\Omega|} - 1\right), \end{aligned} \tag{5.11}$$

where $\frac{|\Omega_{\text{mat}}|}{|\Omega|} = 1 - \frac{|\Omega_{\text{inc}}|}{|\Omega|}$ is the matrix phase volume fraction.

Moreover, there is the differential effective medium approximation which is given in $2D$ by the solution of [1]

$$\left(\frac{\lambda_{\text{inc}} - \lambda_{\text{eff}}}{\lambda_{\text{inc}} - \lambda_{\text{mat}}}\right)\left(\frac{\lambda_{\text{mat}}}{\lambda_{\text{eff}}}\right)^{1/2} = \frac{|\Omega_{\text{mat}}|}{|\Omega|}, \tag{5.12}$$

[3]For instance, $\log(\lambda_{\text{eff}})$ can be employed to compensate for transformations of the form given by Equation (5.8).

which can be solved using a suitable numerical algorithm.

In the case of an FGM model based on Stokes flow, there is no quantity in the governing equations such as conductivity or permeability. Nevertheless, both the Maxwell-Garnett and self-consistent approximations can still be applied since one may consider the infinite contrast limit $\frac{\lambda_{\text{inc}}}{\lambda_{\text{mat}}} \to 0$ which leads to finite quantities in both cases. For the MGA, it is found that

$$\lambda_{\text{eff}} = \lambda_{\text{mat}} \frac{1 - \frac{|\Omega_{\text{inc}}|}{|\Omega|}}{1 + \frac{|\Omega_{\text{inc}}|}{|\Omega|}} \tag{5.13}$$

and equivalently for the SCA

$$\lambda_{\text{eff}} = \lambda_{\text{mat}} \left(1 - 2\frac{|\Omega_{\text{inc}}|}{|\Omega|}\right) \tag{5.14}$$

where obviously the question arises which value to put for λ_{mat}: If the above approximations in the limit of fully insulating spherical inclusions are applied as feature functions of the form $\varphi_{jm}(\boldsymbol{\lambda}_f)$, it is irrelevant which value to put for λ_{mat}, because its value will be absorbed in the corresponding model parameter coefficient $\tilde{\theta}_{c,jm}$.

Generalized means For Darcy flow through binary random media in $1D$, it is a well-known result [1] that the effective permeability λ_{eff} has the closed form solution

$$\lambda_{\text{eff}} = M_{-1}(\boldsymbol{\lambda}_f) = N_{\text{pixels}} \left(\sum_{i=1}^{N_{\text{pixels}}} \lambda_{f,i}^{-1}\right)^{-1} \tag{5.15}$$

where $\boldsymbol{\lambda}_f$ is constructed such that $\lambda_{f,i}$ is the permeability at pixel i of a discretized (one-dimensional) image of the microstructure. The function $M_{-1}(\boldsymbol{\lambda}_f)$ is known as the harmonic mean. Moreover, in 2-dimensional Darcy flow problems through random binary media it has been shown [261] that

$$M_{-1}(\boldsymbol{\lambda}_f) \leq \lambda_{\text{eff}} \leq M_1(\boldsymbol{\lambda}_f) = \frac{1}{N_{\text{pixels}}} \sum_{i=1}^{N_{\text{pixels}}} \lambda_{f,i} \tag{5.16}$$

where $M_1(\boldsymbol{\lambda}_f) \geq M_{-1}(\boldsymbol{\lambda}_f)$ corresponds to the arithmetic mean.

Both the harmonic and arithmetic means are members of a parametric family of functions, namely the generalized mean

$$M_\zeta(\boldsymbol{\lambda}_f) = \left(\frac{1}{N_{\text{pixels}}} \sum_{i=1}^{N_{\text{pixels}}} \lambda_{f,i}^{\zeta}\right)^{1/\zeta} . \tag{5.17}$$

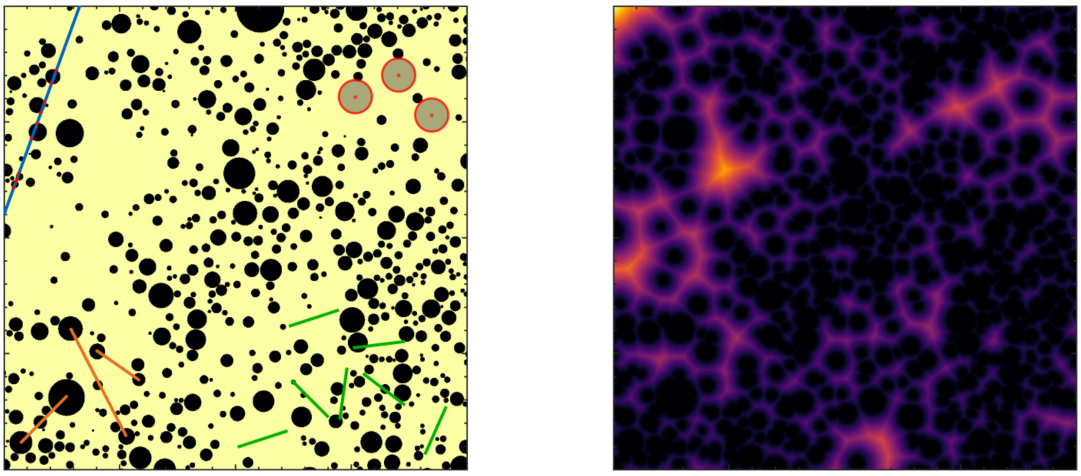


FIGURE 5.4: Graphical representation of certain families of feature functions. The blue line with the red dots in the upper left corner illustrates the *chord length density*, which is the density of the lengths of line segments that lie completely in the solid/fluid phase (separated by the red dots). The red and gray circles in the upper right corner visualize the *pore size density*, which is the density of distances to the closest solid/fluid interface from a random point in the pore space. The green lines in the lower right corner represent both the lineal-path and the two-point correlation functions $L^{(i)}(r, \boldsymbol{\lambda}_f)$, $S_2^{(i)}(r, \boldsymbol{\lambda}_f)$: $L^{(i=i_0)}(r = r_0, \boldsymbol{\lambda}_f)$ gives the probability that a random line segment of length r_0 lies wholly in phase i_0, whereas $S_2^{(i=i+0)}(r = r_0, \boldsymbol{\lambda}_f)$ measures the probability that both end-points of a line segment of length r_0 lie in phase i_0. The orange lines in the lower left corner represent center-to-center mutual exclusion distances. The right picture shows an Euclidean distance transform of the left microstructure can be used as a basis to construct feature functions. Picture taken from [87].

Also, the well-known geometric mean

$$M_0(\boldsymbol{\lambda}_f) = \left(\prod_{i=1}^{N_{\text{pixels}}} \lambda_{f,i} \right)^{1/N_{\text{pixels}}} \tag{5.18}$$

is a member of that family of functions, namely for $\zeta = 0$. It is noted that $M_a(\boldsymbol{\lambda}_f) \geq M_b(\boldsymbol{\lambda}_f)$ for $a > b$. It is thus reasonable to include generalized means M_ζ for $-1 \leq \zeta \leq 1$ as feature functions to the model. Also, generalized means for different values of ζ can be applied, which may turn out to be predictive for the effective material property in conjunction with different feature functions applied. Generalized means can also be evaluated on subregions $\Omega_{\text{sub}} \subset \Omega_m$, e.g., un straight lines in x- or y-direction from edge to edge, as visualized by the green lines in Figure 5.5.

Two-point correlation function The two-point correlation function (also called autocorrelation function) $S_2^{(i)}(r, \boldsymbol{\lambda}_f)$ measures the probability to find two points $\boldsymbol{x}_1, \boldsymbol{x}_2$ with distance $r = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$ both in phase $i \in \{s, f\}$, which can easily be measured either by Monte Carlo or by counting all pixels that are at a distance r apart and both in phase i. Feature functions can be constructed by fixing the phase $i = i_0$ and distance $r = r_0$ such that $\varphi_{jm}(\boldsymbol{\lambda}_f) = S_2^{(i=i_0)}(r = r_0, \boldsymbol{\lambda}_f)$. The two-point correlation is represented graphically by the green lines in Figure 5.4.

Specific surface The specific surface s measures the interface area between solid and fluid phases per unit volume. There are several ways to measure the specific surface: on a pixel-based image of the microstructure $\boldsymbol{\lambda}_f$, one can count the interface pixel edges. Given microstructures with non-overlapping spherical exclusions (see Section 2.4.2), one may simply sum up the circumference of all exclusions. Moreover, it was shown in [266] that in $2D$, the specific surface s is given by $s = -\pi \frac{dS_2^{(i)(r)}}{dr}$.

Kozeny-Carman equation The Kozeny-Carman equation [95–97, 267] gives an estimate for the permeability of a porous microstructure according to

$$\lambda_{\text{eff}} = \frac{(|\Omega_f|/|\Omega|)^3}{5s^2(1 - |\Omega_f|/|\Omega|)^2}, \tag{5.19}$$

where s is the specific surface. Care must be taken for microstructures where $|\Omega_f| \approx |\Omega|$ since in that case the denominator tends to zero and λ_{eff} blows up.

Lineal-path function The lineal-path function $L^{(i)}(r, \boldsymbol{\lambda}_f)$ [256] measures the probability that a randomly dropped line segment of length r lies wholly in phase i. This can be measured either by Monte Carlo or by looping over all pixels of distance r and counting the instances where the above condition is fulfilled. Feature functions can be constructed by fixing the phase $i = i_0$ and distance $r = r_0$ such that $\varphi_{jm}(\boldsymbol{\lambda}_f) = L^{(i=i_0)}(r = r_0, \boldsymbol{\lambda}_f)$. Moreover, for spherical exclusions, it can be shown [1] that $L^{(i)}$ takes the form (in the limit of infinitely large microstructures) $L^{(i)}(r, \boldsymbol{\lambda}_f) = \frac{|\Omega_{i,m}|}{|\Omega_m|} e^{r/\ell}$. Another feature function can be constructed by estimating the length scale ℓ, e.g., by a least squares fit of an exponential decay function to several lineal-path evaluations at different distances. A visualization of the lineal-path function is given by the green lines in Figure 5.4.

Chord length density The chord length density can be understood as follows: given an infinitely long straight line going through an infinitely large random binary microstructure $\boldsymbol{\lambda}_f$, the chord length density $p_{\text{chord}}^{(i)}(r|\boldsymbol{\lambda}_f)$ gives the density of lengths of line segments that go from phase boundary to phase boundary within phase i, see the blue line in Figure 5.4 for a graphical illustration. Feature functions can be constructed, e.g., by computing the mean or other moments of the distribution $p_{\text{chord}}^{(i)}(r|\boldsymbol{\lambda}_f)$. In discretized images of isotropic random media, the chord length density can be estimated by counting connected pixels in phase i on all possible straight lines going in up/down or left/right direction. For the matrix (fluid) phase of spherical exclusion systems, analytical solutions based on the expected exclusion radii and the volume fractions are available [1].

Pore size density The pore size density $p^{(f)}_{\text{pore}}(r|\boldsymbol{\lambda}_f)$ gives the probability density that a randomly chosen point $\boldsymbol{x}$ in the pore (fluid) space Ω_f is at a distance r from the nearest solid/fluid interface. In a discretized image, this can be evaluated by sampling a random point $\boldsymbol{x} \in \Omega_f$ and computing the distances of $\boldsymbol{x}$ to all pixels of the image. The distance to the closest pixel in phase Ω_s gives the radius of the pore space. The pore size density can be approximated by a histogram of the found pore space radii. Approximate analytical expressions exist for microstructures with spherical exclusions [1]. The pore size density is visualized via the red and gray circles in Figure 5.4.

Statistics of exclusion radii For microstructures based on random polydisperse spherical exclusions, features can be constructed from the statistics (e.g., mean, standard deviation, minimum, maximum) of the exclusion radii.

Statistics of mutual exclusion distances In microstructures with random spherical exclusions, feature functions can be designed from the statistics of mutual exclusion distances, measured from edge to edge or center to center. Center-to-center mutual distances are visualized by orange lines in Figure 5.4.

Nearest-neighbor functions For microstructures based on spherical exclusions, the nearest-neighbor density functions [1] $H_V(r|\boldsymbol{\lambda}_f)$, $H_P(r|\boldsymbol{\lambda}_f)$ can be defined. $H_V(r|\boldsymbol{\lambda}_f)$, $H_P(r|\boldsymbol{\lambda}_f)$ give the probability density that at an arbitrary point in the microstructure (for H_V)/at an arbitrary spherical exclusion center (for H_P) the center of the nearest neighbor particle is at a distance r. One can define the same functions h_V, h_P considering the nearest neighbor surface instead of the nearest neighbor center. Approximate analytical expressions exist based on the volume fractions and the average mutual distances of spherical exclusions.

Exclusion probability functions For microstructures with spherical exclusions, the exclusion probability functions $E_V(r|\boldsymbol{\lambda}_f), E_P(r|\boldsymbol{\lambda}_f)$ [1] give the probability of finding a spherical cavity of radius r around an arbitrary point (for E_V)/an arbitrary exclusion center (for E_P) in the microstructure $\boldsymbol{\lambda}_f$. It can be shown that $H_V(r|\boldsymbol{\lambda}_f) = -\frac{\partial E_V(r|\boldsymbol{\lambda}_f)}{\partial r}$, $H_P(r|\boldsymbol{\lambda}_f) = -\frac{\partial E_P(r|\boldsymbol{\lambda}_f)}{\partial r}$ [1]. Accordingly, there exist closed form approximations just as for the nearest neighbor functions H_V, H_P.

Distance transforms Given the binary microstructure as a discretized image of pixels, one can compute the distance transform. The distance transform computes for every pixel in phase i the distance to the closest pixel of phase $\neg i$ and assigns the value to the corresponding pixel of the distance-transformed image, see Figure 5.4. A feature function φ can for instance be based on any statistic of the distance-transformed image, e.g., the mean, standard deviation, maximum, etc.

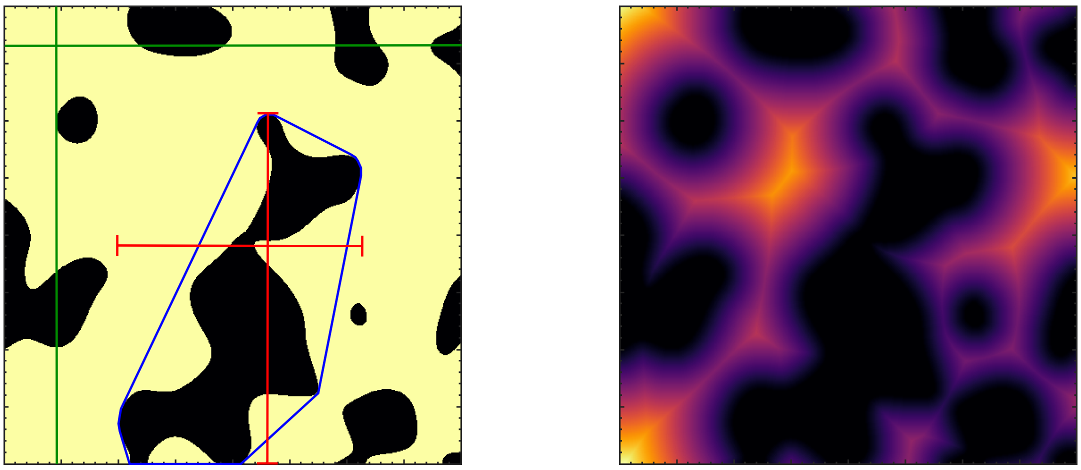


FIGURE 5.5: Visualization of a blob convex area (blue line) and its maximal extension in x- and y-direction. The green lines show possible paths along which generalized means can be taken. The right part of the figure shows a Euclidean distance transform of the microstructure on the left. Picture taken from [103].

Image properties By making use of pre-implemented image analysis functions [268] on the binary random field defining the microstructure $\boldsymbol{\lambda}_f$, one can construct features such as the average size/number/convex area /extent in x- or y-direction of connected phase blobs and any statistics of those quantities. The blob convex area and maximum extension in x-/y-direction is visualized in Figure 5.5.

Many-body potentials For microstructures with spherical exclusions, one may use many-body potentials $V(d)$ as features like

$$\varphi_{jm}(\boldsymbol{\lambda}_f) = \sum_{\substack{\text{exclusion} \\ \text{pairs in } \boldsymbol{\lambda}_f}} V(d_j), \tag{5.20}$$

where d_j is the center-to-center distance of exclusion pair j. Different kinds of potentials like square wells or the Lennard-Jones potential can be applied.

Ising energy Given a discretized image of the microstructure, one may use the energy of a $2D$ Ising spin system, where the solid phase corresponds to 'spin up' and the fluid phase to 'spin down'. The feature function has the form

$$\varphi_{jm}(\boldsymbol{\lambda}_f) = \sum_{\langle i,j \rangle \text{ in } \boldsymbol{\lambda}_f} \lambda_{f,i}\lambda_{f,j}, \tag{5.21}$$

where $\langle i,j \rangle$ in $\boldsymbol{\lambda}_f$ denotes nearest neighbor pixels in $\boldsymbol{\lambda}_f$. The interaction constant can be dropped as it can be absorbed into the corresponding model parameter $\tilde{\theta}_{c,jm}$.

Shortest path from left to right/up to down Estimating an effective material property for a binary material in a square subregion $\Omega_m \subset \Omega$, one could compute the

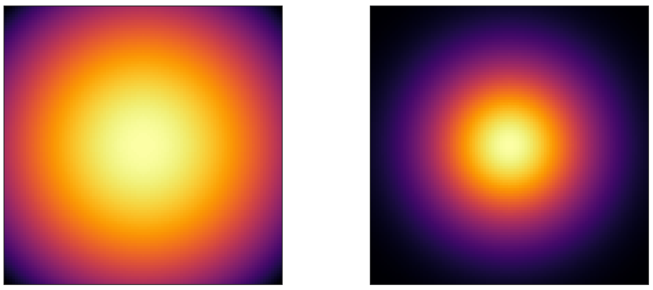


FIGURE 5.6: Visualization of what is called the "Gaussian linear filter" in [104]. The feature function output $\varphi_{jm}(\boldsymbol{\lambda}_f)$ is the inner product of the above Gaussians with the binary microstructure in the square subregion Ω_m.

smallest distance of a path in the fluid phase in left/right up/down direction from square edge to edge[4]. If no connected path through the fluid phase exists, the feature may return the maximum possible distance in the cell, i.e., the square diagonal.

"Gaussian linear filter" To model that pixels close to the center of a square subregion Ω_m should be more important than pixels near the boundary, one can apply what is called a "Gaussian linear filter" in [104], see Figure 5.6. The feature function is given by the inner product of the discretized image of the microstructure with the linear filter.

Autoencoder representations A class of feature functions that can be constructed via semi-supervised learning is given by autoencoder representations [156, 226], which also include PCA loadings. The principle is as follows: based on a large unsupervised set of microstructures $\boldsymbol{\lambda}_f$, train an autoencoder/do a PCA to find a low-dimensional representation $\boldsymbol{\xi}, \dim(\boldsymbol{\xi}) \ll \dim(\boldsymbol{\lambda}_f)$ that is optimal in the sense that it allows to reconstruct $\boldsymbol{\lambda}_f$ with minimal reconstruction error. The components ξ_i may be used as features φ.

5.2.2 The decoder distribution p_{cf}

The coarse-to-fine mapping $\boldsymbol{u}_c \mapsto \boldsymbol{u}_f$ that is mediated by the decoder distribution $p_{cf}(\boldsymbol{u}_c|\boldsymbol{u}_f, \boldsymbol{\theta}_{cf})$ should take into consideration the spatial properties of the problem: as mentioned before, the component $u_{f,i}$ of the FGM response vector is associated with a point $\boldsymbol{x}_i \in \Omega$ by the fact that it corresponds to the FGM response field at that point. In both Stokes and Darcy paradigms, this is the pressure field $P(\boldsymbol{x})$ such that $u_{f,i} = P(\boldsymbol{x}_i)$ where $\boldsymbol{x}_i \in G^{(f)}$ and $G^{(f)}$ is a regular square grid in Ω where P is interpolated on from the FGM solution field.

Similarly, $\boldsymbol{u}_c$ is associated with spatial locations via $u_{c,j} = \bar{P}(\boldsymbol{x}_j), \boldsymbol{x}_j \in G^{(c)}$, where $\bar{P}$ is the pressure response of the CGM Darcy model (see Section 2.2) and $G^{(c)}$ is a square grid in Ω of much coarser shape than $G^{(f)}$.

[4]This is done with the Matlab built-in `bwdistgeodesic` function, see [93].

Given this spatial structure of the problem, it is reasonable to construct p_{cf} in the form of an interpolator in the sense that components $u_{c,j}$ associated with $\boldsymbol{x}_j$ should contribute more to components $u_{f,i}$ associated with $\boldsymbol{x}_i$ the smaller the distance $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$. Given that the Darcy-type CGM finite element solution is of the form (see Section 3.1.1)

$$u_c(\boldsymbol{x}) = \bar{P}(\boldsymbol{x}) \sum_{j=1}^{\dim(\boldsymbol{u}_c)} u_{c,j}\psi_j(\boldsymbol{x}), \tag{5.22}$$

it appears natural to adopt a model of the form

$$\begin{aligned} u_{f,i} = P(\boldsymbol{x}_i) &= u_c(\boldsymbol{x}_i) + \tau_{cf,i}^{-1/2} Z_i \qquad\qquad Z_i \sim \mathcal{N}(0,1) \\ &= \sum_{j=1}^{\dim(\boldsymbol{u}_c)} u_{c,j}\psi_j(\boldsymbol{x}_i) + \tau_{cf,i}^{-1/2} Z_i = \boldsymbol{W}\boldsymbol{u}_c + \tau_{cf,i}^{-1/2} Z_i, \end{aligned} \tag{5.23}$$

where $\boldsymbol{W}$ plays the role of an interpolation matrix from $G^{(c)}$ to $G^{(f)}$ with the components $W_{ij} = \psi_j(\boldsymbol{x}_i)$, where ψ_j are the CGM shape functions and $\tau_{cf,i}$ is the precision of component $u_{f,i}$. Alternatively to fixing $W_{ij} = \psi_j(\boldsymbol{x}_i)$, it is possible to treat $\boldsymbol{W}$ as a free parameter to be learned from the data. It is noted though that $\boldsymbol{W} \in \mathbb{R}^{\dim(\boldsymbol{u}_f)\times\dim(\boldsymbol{u}_c)}$, i.e., the high number of free parameters in $\boldsymbol{W}$ need to be reduced by either introducing further constraints or strong priors need to be applied in order to avoid overfitting. Finally, the decoder distribution p_{cf} is given by

$$p_{cf}(\boldsymbol{u}_f|\boldsymbol{u}_c(\boldsymbol{\lambda}_c), \boldsymbol{\theta}_{cf}) = \mathcal{N}(\boldsymbol{u}_f|\boldsymbol{W}\boldsymbol{u})c(\boldsymbol{\lambda}_c), \mathrm{diag}(\boldsymbol{\tau}_{cf}^{-1})) \tag{5.24}$$

with model parameters $\boldsymbol{\theta}_{cf} = \{\boldsymbol{W}, \boldsymbol{\tau}_{cf}\}$.

An interesting alternative for modeling the decoder p_{cf} is given by Bayesian nonparametric models such as Gaussian process (GP) regression [47, 155] from $u_{c,j} = \bar{P}(\boldsymbol{x}_j)$ to $u_{f,i} = P(\boldsymbol{x}_i)$. GP regression can exploit the spatial structure of the problem by employing a stationary covariance kernel of the form

$$\mathrm{cov}(u_{c,j}, u_{f,i}) = \mathrm{cov}(\bar{P}(\boldsymbol{x}_j), P(\boldsymbol{x}_i)) = \mathrm{cov}(\boldsymbol{x}_j, \boldsymbol{x}_i) = \mathrm{cov}(\boldsymbol{x}_j - \boldsymbol{x}_i).$$

5.3 Sparsity prior models and model training

It is clear from the discussion in Section 5.2.1 that the number of feature functions applied in the encoder distribution p_c is practically only limited by computational resources, see Section 5.5. As it is typically unknown a priori which microstructural features are important/unimportant for a certain effective material property, one strategy can be to sequentially add features from a predefined [148, 269, 270] or parametric [201, 271] family. The strategy advocated in this work is to include a large, wide-ranging library of feature functions $\varphi_{jm}(\boldsymbol{\lambda}_f)$ from the beginning and let a sparsity prior filter out the most relevant features based on the training data.

It is clear that the more feature functions $\varphi_{jm}(\boldsymbol{\lambda}_f)$ are included, the more free model parameters $\tilde{\theta}_{jm}$ are to be learned from the data. As p_c is specified as a Gaussian linear model, finding a maximum-likelihood estimate[5] for the encoder parameters $\tilde{\boldsymbol{\theta}}_c$ is possible but would end up in severe overfitting due to excessive model complexity. A common remedy in the Bayesian framework is the use of prior distributions to regularize overly complex models.

In particular, it is desirable to use priors that enforce *sparsity*, i.e., priors that prune all but a handful $\tilde{\theta}_{c,jm}$'s which correspond to feature functions that, based on the training data, appear to be most predictive for the sought effective material property. During the development of this work, several different types of sparsity priors have been applied/designed for the latent variable model defined by Equation (5.3), such as Student-t type priors [147, 150, 154] (see Section 4.2.2), Laplacian priors (or LASSO regression [145, 272], see Section 4.2.2, published in [103]), the relevance vector machine [153, 156] (see Section 4.2.2, published in [104]) and the variational relevance vector machine [161, 162] (see Section 4.2.2, published in [87]).

Not only will the application of one of the above sparsity priors prevent the model from overfitting, but as well allow for physical interpretation as it extracts the microstructural features that are most significant for prediction of effective material properties. Additionally and more importantly, sparseness in $\tilde{\boldsymbol{\theta}}_c$-space leads to computational benefits in the prediction stage as pruned features are not required to be evaluated anymore.

5.3.1 The maximum likelihood approach

In this subsection, we describe the model training by maximizing the likelihood function $\mathcal{L}(\mathcal{D}|\boldsymbol{\theta}_{cf}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c)$ for the training data $\mathcal{D} = \left\{\boldsymbol{\lambda}_f^{(n)}, \boldsymbol{u}_f^{(n)}\right\}_{n=1}^N$. Obviously, as explained in the previous paragraph, a pure maximum likelihood approach would be incompatible with the discussed desiderata of including as many microstructural feature functions $\varphi_{jm}(\boldsymbol{\lambda}_f)$ (and therefore parameters $\tilde{\theta}_{jm}$) as possible. This subsection can thus be seen as purely academic with the intention to set the basis and clarify the notation for the subsequent discussion of sparsity enforcing models.

According to Equation (5.3) and assuming independent and identically distributed samples, the likelihood function for the training data $\mathcal{D}$ is given by

$$\begin{aligned}\mathcal{L}(\mathcal{D}|\boldsymbol{\theta}_{cf}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c) &= \prod_{n=1}^N p(\boldsymbol{u}_f^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \boldsymbol{\theta}_{cf}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c) \\ &= \prod_{n=1}^N \int p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\theta}_{cf}) p_c(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c) d\boldsymbol{\lambda}_c^{(n)}.\end{aligned} \tag{5.25}$$

[5]For given decoder parameters $\boldsymbol{\theta}_{cf}$

Model training

The latent variable model defined by Equation (5.3) can be trained efficiently by the *Expectation-Maximization* (EM) algorithm [174], see Section 4.3.1. In order to do so, we apply Jensen's inequality to find the lower bound

$$\begin{aligned}
\log \mathcal{L}(\mathcal{D}|\boldsymbol{\theta}_{cf}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c) &= \sum_{n=1}^{N} \int p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\theta}_{cf}) p_c(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c) d\boldsymbol{\lambda}_c^{(n)} \\
&\geq \int q_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}) \log \frac{p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\tau}_{cf}) p_c(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c)}{q_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)})} d\boldsymbol{\lambda}_c^{(n)} \\
&= \sum_{n=1}^{N} \mathcal{F}^{(n)}\left(\left\{q_{\boldsymbol{\lambda}_c^{(n)}}\right\}_{n=1}^{N}; \boldsymbol{\theta}_{cf}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c\right) = \mathcal{F}\left(\left\{q_{\boldsymbol{\lambda}_c^{(n)}}\right\}_{n=1}^{N}; \boldsymbol{\theta}_{cf}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c\right)
\end{aligned} \tag{5.26}$$

where the $q_{\boldsymbol{\lambda}_c^{(n)}}$ can be arbitrary probability densities. As discussed in Section 4.3.1, the EM-algorithm iteratively computes expected values and maximizes the lower bound $\mathcal{F}$ w.r.t. the auxiliary distributions $\left\{q_{\boldsymbol{\lambda}_c^{(n)}}\right\}_{n=1}^{N}$ (E-step), and subsequently maximizes $\mathcal{F}$ w.r.t. the model parameters $\boldsymbol{\theta}_{cf}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c$ (M-step). It is easily seen that the auxiliary distribution $q_{\boldsymbol{\lambda}_c^{(n)}}$ that maximizes the lower bound $\mathcal{F}^{(n)}$ for a given set of parameters $\boldsymbol{\theta}_{cf}^{(t)}, \tilde{\boldsymbol{\theta}}_c^{(t)}, \boldsymbol{\tau}_c^{(t)}$ is given by

$$q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}(\boldsymbol{\lambda}_c^{(n)}) \propto p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\tau}_{cf}^{(t)}) p_c(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tilde{\boldsymbol{\theta}}_c^{(t)}, \boldsymbol{\tau}_c^{(t)}) \tag{5.27}$$

because this is the equality limit of the inequality in Equation (5.26).

To maximize $\mathcal{F}$ w.r.t. the parameters $\boldsymbol{\theta}_{cf}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c$ for fixed auxiliary distributions as given by Equation (5.27), it is necessary to compute gradients w.r.t. the parameters,

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{F} = \nabla_{\boldsymbol{\theta}} \sum_{n=1}^{N} \Bigg(&\left\langle \log p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\tau}_{cf}) \right\rangle_{q_{\lambda_c^{(n)}}^{(t)}} \\
&+ \left\langle \log p_c(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c) \right\rangle_{q_{\lambda_c^{(n)}}^{(t)}} \Bigg)
\end{aligned} \tag{5.28}$$

where $\boldsymbol{\theta}$ denotes an arbitrary model parameter. Elementary mathematics and the previous definitions for p_c, p_{cf} yield the equations

$$\nabla_{\boldsymbol{W}} \mathcal{F}^{(n)} = \operatorname{diag}(\boldsymbol{\tau}_{cf}) \left(\boldsymbol{u}_f^{(n)} \langle \boldsymbol{u}_c^T(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q^{(t)}_{\lambda_c^{(n)}}} - \boldsymbol{W} \langle \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}) \boldsymbol{u}_c^T(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q^{(t)}_{\lambda_c^{(n)}}} \right), \tag{5.29}$$

$$\nabla_{\tau_{cf,i}} \mathcal{F}^{(n)} = \frac{1}{2\tau_{cf,i}} - \frac{1}{2} \left((u_{f,i}^{(n)})^2 - 2u_{f,i}^{(n)} [\boldsymbol{W} \langle \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q^{(t)}_{\lambda_c^{(n)}}}]_i + \langle [\boldsymbol{W} \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)})]_i^2 \rangle_{q^{(t)}_{\lambda_c^{(n)}}} \right), \tag{5.30}$$

$$\nabla_{\tilde{\boldsymbol{\theta}}_{c,m}} \mathcal{F}^{(n)} = \tau_{c,m} \left(\langle \lambda_{c,m}^{(n)} \rangle_{q^{(t)}_{\lambda_c^{(n)}}} \boldsymbol{\varphi}_m^{(n)} - \left(\boldsymbol{\varphi}_m^{(n)} (\boldsymbol{\varphi}_m^{(n)})^T \right) \tilde{\boldsymbol{\theta}}_{c,m} \right), \tag{5.31}$$

$$\nabla_{\tau_{c,m}} \mathcal{F}^{(n)} = \frac{1}{2\tau_{c,m}} - \frac{1}{2} \left(\langle (\lambda_{c,m}^{(n)})^2 \rangle_{q^{(t)}_{\lambda_c^{(n)}}} - 2 \langle \lambda_{c,m}^{(n)} \rangle_{q^{(t)}_{\lambda_c^{(n)}}} (\boldsymbol{\varphi}_M^{(n)})^T \tilde{\boldsymbol{\theta}}_{c,m} + ((\boldsymbol{\varphi}_m^{(n)})^T \tilde{\boldsymbol{\theta}}_{c,m})^2 \right), \tag{5.32}$$

with $\boldsymbol{\varphi}_m^{(n)} = \boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f^{(n)})$. To find the maximum likelihood parameter estimates, the above gradients can be fed to any gradient-based (stochastic) optimizer or, as all model parameters only appear up to linear order, closed-form update equations can be obtained by setting the gradients to 0, given estimates of the involved expected values. It is readily seen that the matrix $\sum_{n=1}^N \boldsymbol{\varphi}_m^{(n)} (\boldsymbol{\varphi}_m^{(n)})^T$ is rank deficient if $\dim(\boldsymbol{\varphi}_m^{(n)}) = \dim(\tilde{\boldsymbol{\theta}}_{c,m}) < N$. In such a case, there would exist an infinite number of solutions $\tilde{\boldsymbol{\theta}}_{c,m}^*$ which all perfectly reconstruct the latent $\lambda_{c,m}^{(n)}$'s – a signal of severe overfitting.

The expected values $\langle \lambda_{c,m}^{(n)} \rangle_{q^{(t)}_{\lambda_c^{(n)}}}$, $\langle (\lambda_{c,m}^{(n)})^2 \rangle_{q^{(t)}_{\lambda_c^{(n)}}}$, $\langle u_{c,k}(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q^{(t)}_{\lambda_c^{(n)}}}$, $\langle \boldsymbol{u}_{c,k}^2(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q^{(t)}_{\lambda_c^{(n)}}}$ w.r.t. $q^{(t)}_{\lambda_c^{(n)}}$ involve the CGM solver $\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)})$ such that they are not given in closed form and need to be estimated using one of the approximate inference methods presented in Section 4.3. As gradients $\nabla_{\lambda_c} \boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ are available (see Section 3.1.1), it is suggested to use a gradient-based MCMC kernel (as, e.g., presented in Section 11) or stochastic variational inference with reparametrization trick (see Section 4.3.4).

5.3.2 The Laplacian prior model

In the work published in [103], a Laplacian prior of the form

$$p(\tilde{\boldsymbol{\theta}}_c | \beta) = \frac{\sqrt{\beta}}{2} \exp \left\{ -\sqrt{\beta} \sum_{m=1}^{\dim(\lambda_c)} \sum_{j=1}^{N_{\text{features},m}} \left| \tilde{\theta}_{c,jm} \right| \right\} \tag{5.33}$$

is adopted on a surrogate model for Darcy flow simulation which is calibrated by finding MAP estimates of the model parameters. As discussed in Section 4.2.2, a Laplacian prior leads to sparse MAP estimates for $\tilde{\boldsymbol{\theta}}_c$ where the degree of sparsity is controlled by the hyperparameter β. Appropriate values for β can be estimated, e.g., via cross-validation [155, 273], the Akaike information criterion [274, 275], or by minimization of Stein's unbiased risk estimate [149, 276].

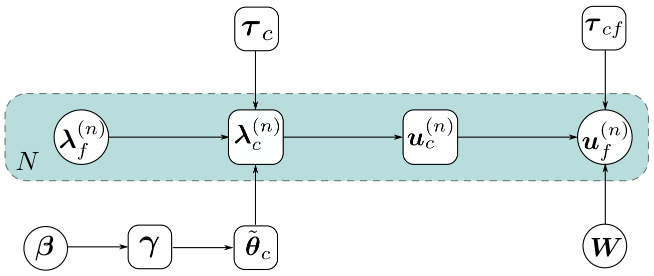


FIGURE 5.7: Graphical representation of the Bayesian network defined by the posterior given in Equation (5.37). The part with the teal background corresponds to the likelihood and is repeated N-times.

A characteristic pitfall of the Laplacian prior is that MAP estimates, despite being unique[6], can not be found in closed form/via gradient-based optimization since the posterior is not differentiable for points where any component $\tilde{\theta}_{c,jm} = 0$ such that gradients typically do not vanish at the optimum. One valid approach to maximize the posterior in $\tilde{\boldsymbol{\theta}}_c$ is given by the EM-algorithm [147] (see Section 4.3.1) which is discussed in the sequel.

Model training

For efficiency and modeling purposes, it is assumed that the coarse-to-fine interpolation matrix $\boldsymbol{W}$ is fixed to $W_{ij} = \psi_j(\boldsymbol{x}_i)$ (as discussed in Section 5.2.2) such that $\boldsymbol{\theta}_{cf} = \boldsymbol{\tau}_{cf}$. We further assume a prior $p(\boldsymbol{\tau}_{cf}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c)$ of the form

$$p(\boldsymbol{\tau}_{cf}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c) = p(\tilde{\boldsymbol{\theta}}_c|\beta) \cdot \text{const.}, \tag{5.34}$$

with $p(\tilde{\boldsymbol{\theta}}_c|\beta)$ given by the Laplacian in Equation (5.33). The crucial idea in maximizing the posterior in $\tilde{\boldsymbol{\theta}}_c$ via EM is to write the Laplacian prior as

$$p(\tilde{\boldsymbol{\theta}}_c|\beta) = \prod_{jm} \int_0^{\infty} \mathcal{N}(\tilde{\theta}_{c,jm}|0, \gamma_{jm}) p(\gamma_{jm}|\beta) d\gamma_{jm}, \tag{5.35}$$

where $p(\gamma_{jm}|\beta)$ is

$$p(\gamma_{jm}|\beta) = \frac{\beta}{2} \exp\left\{-\frac{\beta}{2}\gamma_{jm}\right\}, \qquad \gamma_{jm} \geq 0. \tag{5.36}$$

The posterior $p(\tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c, \boldsymbol{\tau}_{cf}|\mathcal{D})$ is then given by

$$\begin{aligned} p(\tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c, \boldsymbol{\tau}_{cf}|\mathcal{D}) &\propto \mathcal{L}(\mathcal{D}|\boldsymbol{\tau}_{cf}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c) p(\tilde{\boldsymbol{\theta}}_c|\beta) \\ &= \int \prod_{n=1}^{N} p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\tau}_{cf}) p_c(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c) d\boldsymbol{\lambda}_c^{(n)} \\ &\quad \cdot \prod_{jm} \mathcal{N}(\tilde{\theta}_{c,jm}|0, \gamma_{jm}) p(\gamma_{jm}|\beta) d\gamma_{jm} \end{aligned} \tag{5.37}$$

[6]Assuming a Gaussian likelihood and fixed training data

which defines a Bayesian graphical network depicted in Figure 5.7. Using Jensen's inequality and the conditional independence of $\boldsymbol{\lambda}_c^{(n)}$ and γ_{jm}, this can be lower bounded as

$$\begin{aligned}\log p(\tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c, \boldsymbol{\tau}_{cf}|\mathcal{D}) \geq & \sum_{n=1}^{N} \int q_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}) \log \frac{p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\tau}_{cf}) p_c(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c)}{q_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)})} d\boldsymbol{\lambda}_c^{(n)} \\ & + \sum_{jm} \int q_{\gamma_{jm}}(\gamma_{jm}) \log \frac{\mathcal{N}(\tilde{\theta}_{c,jm}|0, \gamma_{jm}) p(\gamma_{jm}|\beta)}{q_{\gamma_{jm}}(\gamma_{jm})} d\gamma_{jm} \\ = & \mathcal{F}(\{q_{\boldsymbol{\lambda}_c^{(n)}}\}_{n=1}^N; \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c, \boldsymbol{\tau}_{cf}) + \mathcal{F}_\gamma(\{q_{\gamma_{jm}}\}_{\forall jm}; \tilde{\boldsymbol{\theta}}_c, \gamma).\end{aligned} \tag{5.38}$$

Due to factorization, $q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}$ is again given by Equation (5.27) whereas the optimal $q_{\gamma_{jm}}^{(t)}$ is

$$q_{\gamma_{jm}}^{(t)}(\gamma_{jm}) \propto \mathcal{N}(\tilde{\theta}_{c,jm}^{(t)}|0, \gamma_{jm}) p(\gamma_{jm}|\beta). \tag{5.39}$$

Gradients w.r.t. $\boldsymbol{\tau}_{cf}$, $\boldsymbol{\tau}_c$ are identical to the ones given by Equations (5.30), (5.32) and the closed-form updates are given by

$$(\tau_{cf,i}^{(t+1)})^{-1} = \sum_{n=1}^{N} \left((u_{f,i}^{(n)})^2 - 2u_{f,i}^{(n)} [\boldsymbol{W} \langle \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}}]_i + \langle [\boldsymbol{W}\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)})]_i^2 \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}} \right), \tag{5.40}$$

$$(\tau_{c,m}^{(t+1)})^{-1} = \sum_{n=1}^{N} \left(\langle (\lambda_{c,m}^{(n)})^2 \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}} - 2 \langle \lambda_{c,m}^{(n)} \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}} \boldsymbol{\varphi}_m^{(n)} \cdot \tilde{\boldsymbol{\theta}}_{c,m} + (\boldsymbol{\varphi}_m^{(n)} \cdot \tilde{\boldsymbol{\theta}}_{c,m})^2 \right). \tag{5.41}$$

Computing zeros of the gradient $\nabla_{\tilde{\boldsymbol{\theta}}_m}(\mathcal{F} + \mathcal{F}_\gamma)$ yields the update equation for $\tilde{\boldsymbol{\theta}}_{c,m}$,

$$\tilde{\boldsymbol{\theta}}_{c,m}^{(t+1)} = \left(\sum_{n=1}^{N} \boldsymbol{\varphi}_m^{(n)} (\boldsymbol{\varphi}_m^{(n)})^T + \frac{\langle \operatorname{diag}(\boldsymbol{\gamma}_m) \rangle_{q_{\gamma m}^{(t)}}}{\tau_{c,m}^{(t)}} \right)^{-1} \sum_{n'=1}^{N} \left\langle \lambda_{c,m}^{(n')} \right\rangle_{q_{\boldsymbol{\lambda}_c^{(n')}}^{(t)}} \boldsymbol{\varphi}_m^{(n')}, \tag{5.42}$$

where elementary integration gives

$$\left\langle \gamma_{jm} \right\rangle_{q_{\gamma_{jm}}^{(t)}} = \frac{\int \gamma_{jm} \mathcal{N}(\tilde{\theta}_{c,jm}^{(t)}|0, \gamma_{jm}) \frac{\beta}{2} \exp\left\{-\frac{\beta}{2}\gamma_{jm}\right\} d\gamma_{jm}}{\int \mathcal{N}(\tilde{\theta}_{c,jm}^{(t)}|0, \gamma_{jm}) \frac{\beta}{2} \exp\left\{-\frac{\beta}{2}\gamma_{jm}\right\} d\gamma_{jm}} = \frac{\beta}{\left|\tilde{\theta}_{c,jm}^{(t)}\right|}. \tag{5.43}$$

In summary, the E-step is given by estimating/computing the expected values $\langle \lambda_{c,m}^{(n)} \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}}$, $\langle (\lambda_{c,m}^{(n)})^2 \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}}$, $\langle u_{c,k}(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}}$, $\langle u_{c,k}^2(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}}$, and $\left\langle \gamma_{jm} \right\rangle_{q_{\gamma m}^{(t)}}$ via the equations (5.27), (5.39), (5.43) and a suitable approximate inference technique. The M-step comprises the parameter updates for $\tau_{cf,i}^{(t+1)}$, $\tau_{c,m}^{(t+1)}$, and $\tilde{\boldsymbol{\theta}}_{c,m}^{(t+1)}$ as given by equations (5.40), (5.41), and (5.42). These steps are run iteratively until convergence to the MAP estimate $\boldsymbol{\tau}_{cf}^*, \boldsymbol{\tau}_c^*, \tilde{\boldsymbol{\theta}}_c^*$ which is sparse in $\tilde{\boldsymbol{\theta}}_{c,m}^*$. The training procedure is summarized in Algorithm 4.

Algorithm 4 : The Laplacian prior model training, see [103, 147].

```
Input : $\boldsymbol{\tau}_{cf}^{(0)}, \boldsymbol{\tau}_c^{(0)}, \gamma^{(0)}, \beta$ ;                // Initialization
1  Evaluate all feature functions $\boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f^{(n)})$;
2  $t \leftarrow 0$
3  while (not converged) do
4      E-step: // Completely parallelizable in $N$
5      for $n = 1$ to $N$ do
6          Update $q_{\boldsymbol{\lambda}_c^{(n)}}^{(t+1)}$ via Equation (5.27)
7          Estimate $\langle \lambda_{c,m}^{(n)} \rangle_{q^{(t)}_{\boldsymbol{\lambda}_c^{(n)}}}$, $\langle (\lambda_{c,m}^{(n)})^2 \rangle_{q^{(t)}_{\boldsymbol{\lambda}_c^{(n)}}}$, $\langle u_{c,k}(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q^{(t)}_{\boldsymbol{\lambda}_c^{(n)}}}$, $\langle u_{c,k}^2(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q^{(t)}_{\boldsymbol{\lambda}_c^{(n)}}}$
            with a suitable approximate inference method
8      end
9      Compute $\langle \gamma_{jm} \rangle_{q_{\gamma_{jm}}^{(t)}}$ using equation (5.43)
10     M-step:
11     Find $\boldsymbol{\tau}_{cf}^{(t+1)}, \boldsymbol{\tau}_c^{(t+1)}, \tilde{\boldsymbol{\theta}}_{c,m}^{(t+1)}$ using the update equations (5.40)–(5.42)
12     $t \leftarrow t + 1$;
13 end
14 return $\boldsymbol{\tau}_{cf}^*, \boldsymbol{\tau}_c^*, p(\tilde{\boldsymbol{\theta}}_c|\mathcal{D}) = \mathcal{N}(\tilde{\boldsymbol{\theta}}_{c,m}|\boldsymbol{\mu}^*_{\tilde{\boldsymbol{\theta}}_{c,m}}, \boldsymbol{\Sigma}^*_{\tilde{\boldsymbol{\theta}}_{c,m}})$
```

5.3.3 The relevance vector machine prior model

The work published in [104] suggests a prior model that is based on the *relevance vector machine* (RVM) [153, 156, 157] adapted to the latent target variables $\boldsymbol{\lambda}_c^{(n)}$. A central advantage of the RVM over the Laplacian prior discussed in the previous subsection is that the RVM does not require any hyperparameter tuning. Moreover, we obtain (approximate) posterior distributions instead of MAP estimates, enabling the quantification and propagation of uncertainties in the model parameters due to limited training data.

Similar to Equation (5.35), we assume a zero-mean Gaussian prior on the feature coefficients $\tilde{\theta}_{jm}$,

$$p(\tilde{\boldsymbol{\theta}}_c, |\boldsymbol{\gamma}) = \prod_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \prod_{j=1}^{N_{\text{features}, m}} \mathcal{N}(\tilde{\theta}_{c,jm}|0, \gamma_{jm}), \tag{5.44}$$

where the γ_{jm}'s are a set of non-negative hyperparameters describing the prior variance for $\tilde{\theta}_{c,jm}$. The basic principle behind RVM is to compute the evidence/marginal likelihood by integration over the model parameters $\tilde{\boldsymbol{\theta}}_c$, and subsequently maximize the evidence w.r.t. the hyperparameters $\boldsymbol{\gamma}$, a strategy which is found under the names "type-II maximum likelihood" or "evidence approximation" in the literature. Again, the precision parameters $\boldsymbol{\tau}_{cf}, \boldsymbol{\tau}_c$ are found via ML, whereas $\boldsymbol{W}$ is fixed to be the CGM shape function interpolant $W_{ij} = \psi_j(\boldsymbol{x}_i)$.

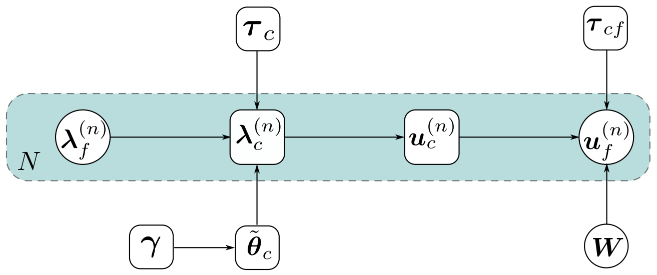


FIGURE 5.8: Graphical representation of the Bayesian network defined by the posterior given in Equation (5.45). The part with the teal background corresponds to the likelihood and is repeated N-times.

Model training

Given the likelihood function in Equation (5.25), the marginal likelihood w.r.t. $\tilde{\boldsymbol{\theta}}_c$ is

$$\begin{aligned}\mathcal{P}(\gamma) &= \prod_{n=1}^{N} \int p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\theta}_{cf}) p_c(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c) d\boldsymbol{\lambda}_c^{(n)} p(\tilde{\boldsymbol{\theta}}_c|\gamma) d\tilde{\boldsymbol{\theta}}_c \\ &= \int \mathcal{L}(\tilde{\boldsymbol{\theta}}_c) p(\tilde{\boldsymbol{\theta}}_c|\gamma) d\tilde{\boldsymbol{\theta}}_c\end{aligned} \tag{5.45}$$

where we drop the dependence of $\mathcal{L}$ on the data $\mathcal{D}$ and the parameters $\boldsymbol{\theta}_{cf}, \boldsymbol{\tau}_c$ for convenience. The corresponding Bayesian graphical model is depicted in Figure 5.8. The value of the hyperparameters γ is determined by

$$\gamma^* = \arg\max_{\gamma} \mathcal{P}(\gamma). \tag{5.46}$$

The latent variable optimization problem defined by Equation (5.46) can again be optimized by the EM algorithm. To do so, we apply Jensen's inequality to find the log evidence lower bound

$$\begin{aligned}\log \mathcal{P}(\gamma) &= \log \int \mathcal{L}(\tilde{\boldsymbol{\theta}}_c) p(\tilde{\boldsymbol{\theta}}_c|\gamma) d\tilde{\boldsymbol{\theta}}_c \\ &\geq \int q_{\tilde{\boldsymbol{\theta}}_c}(\tilde{\boldsymbol{\theta}}_c) \log \frac{\mathcal{L}(\tilde{\boldsymbol{\theta}}_c) p(\tilde{\boldsymbol{\theta}}_c|\gamma)}{q_{\tilde{\boldsymbol{\theta}}_c}(\tilde{\boldsymbol{\theta}}_c)} d\tilde{\boldsymbol{\theta}}_c = \mathcal{G}\left(q_{\tilde{\boldsymbol{\theta}}_c}; \gamma\right),\end{aligned} \tag{5.47}$$

where $q_{\tilde{\boldsymbol{\theta}}_c}$ can again be any arbitrary probability distribution. The optimal $q_{\tilde{\boldsymbol{\theta}}_c}^{(t+1)}$ that maximizes the lower bound $\mathcal{G}$ for a given estimate $\gamma^{(t)}$ is given by

$$q_{\tilde{\boldsymbol{\theta}}_c}^{(t+1)}(\tilde{\boldsymbol{\theta}}_c) \propto \mathcal{L}(\tilde{\boldsymbol{\theta}}_c) p(\tilde{\boldsymbol{\theta}}_c|\gamma^{(t)}) \tag{5.48}$$

as this is the equality limit for the inequality in Equation (5.47). To maximize the expected value defined by Equation (5.47) w.r.t. γ for a given $q_{\tilde{\boldsymbol{\theta}}_c}^{(t+1)}$, it sufficient to exclusively keep terms that depend directly on γ,

$$\mathcal{G}\left(q_{\tilde{\boldsymbol{\theta}}_c}; \gamma\right) \propto -\frac{1}{2} \sum_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \sum_{j=1}^{N_{\text{features},m}} \left(\log \gamma_{jm} + \gamma_{jm}^{-1} \left\langle \tilde{\theta}_{c,jm}^2 \right\rangle_{q_{\tilde{\boldsymbol{\theta}}_c}^{(t+1)}} \right). \tag{5.49}$$

To find the update equation for $\gamma_{jm}^{(t+1)}$, we compute the zeroes of the derivatives $\frac{\partial}{\partial \gamma_{jm}} \mathcal{G}$ to find

$$\gamma_{jm}^{(t+1)} = \left\langle \tilde{\theta}_{c,jm}^2 \right\rangle_{q_{\tilde{\boldsymbol{\theta}}_c}^{(t)}}. \tag{5.50}$$

Assuming fixed parameter estimates $\boldsymbol{\tau}_c^{(t)}$, $\boldsymbol{\tau}_{cf}^{(t)}$ and given the Gaussian linear model for p_c as defined in Equation (5.5), the optimal $q_{\tilde{\boldsymbol{\theta}}_c}^{(t+1)}$ as given in Equation (5.48) is a Gaussian

$$q_{\tilde{\boldsymbol{\theta}}_c}^{(t+1)}(\tilde{\boldsymbol{\theta}}_c) = \prod_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \mathcal{N}(\tilde{\boldsymbol{\theta}}_{c,m} | \boldsymbol{\mu}_{\tilde{\boldsymbol{\theta}}_{c,m}}^{(t+1)}, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}_{c,m}}^{(t+1)}) \tag{5.51}$$

where the means $\boldsymbol{\mu}_{\tilde{\boldsymbol{\theta}}_{c,m}}^{(t+1)}$ are found by maximizing $q_{\tilde{\boldsymbol{\theta}}_c}^{(t+1)}(\tilde{\boldsymbol{\theta}}_c)$ w.r.t. $\tilde{\boldsymbol{\theta}}_{c,m}$,

$$\boldsymbol{\mu}_{\tilde{\boldsymbol{\theta}}_{c,m}}^{(t+1)} = \arg\max_{\tilde{\boldsymbol{\theta}}_{c,m}} q_{\tilde{\boldsymbol{\theta}}_c}^{(t+1)}(\tilde{\boldsymbol{\theta}}_c) = \arg\max_{\tilde{\boldsymbol{\theta}}_{c,m}} \left(\log \mathcal{L}(\tilde{\boldsymbol{\theta}}_c) + \log p(\tilde{\boldsymbol{\theta}}_c | \boldsymbol{\gamma}^{(t)}) \right). \tag{5.52}$$

This maximization problem can be solved jointly with the maximization w.r.t. $\boldsymbol{\tau}_{cf}, \boldsymbol{\tau}_c$ with an inner loop EM procedure with the identical update equations as given in Equation (5.40)–(5.42). It is noted that it is unnecessary to run this maximization to full convergence – even only a single iteration is sufficient.

The covariance parameter $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}_{c,m}}^{(t+1)}$ is determined by the negative inverse Hessian of $\log q_{\tilde{\boldsymbol{\theta}}_c}^{(t+1)}(\tilde{\boldsymbol{\theta}}_c)$,

$$(\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}_{c,m}}^{(t+1)})^{-1} = -\nabla_{\tilde{\boldsymbol{\theta}}_{c,m}} \nabla_{\tilde{\boldsymbol{\theta}}_{c,m}} \log q_{\tilde{\boldsymbol{\theta}}_c}^{(t+1)}(\tilde{\boldsymbol{\theta}}_c) = \tau_{c,m} \sum_{n=1}^{N} \boldsymbol{\varphi}_m^{(n)} (\boldsymbol{\varphi}_m^{(n)})^T + (\mathrm{diag}(\boldsymbol{\gamma}_m^{(t)}))^{-1}. \tag{5.53}$$

The update equation for $\gamma_{jm}^{(t+1)}$ is thus

$$\gamma_{jm}^{(t+1)} = \left\langle \tilde{\theta}_{c,jm}^2 \right\rangle_{q_{\tilde{\boldsymbol{\theta}}_c}^{(t)}} = (\mu_{\tilde{\boldsymbol{\theta}}_{c,m},j}^{(t+1)})^2 + \Sigma_{\tilde{\boldsymbol{\theta}}_{c,m},jj}^{(t+1)}. \tag{5.54}$$

After convergence of $\boldsymbol{\gamma}, \boldsymbol{\mu}_{\tilde{\boldsymbol{\theta}}_{c,m}}, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}_{c,m}}$ to $\boldsymbol{\gamma}^*, \boldsymbol{\mu}_{\tilde{\boldsymbol{\theta}}_{c,m}}^*, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}_{c,m}}^*$, the posterior on the model parameters $\tilde{\boldsymbol{\theta}}_{c,m}$ is given by

$$p(\tilde{\boldsymbol{\theta}}_{c,m} | \mathcal{D}) = \mathcal{N}(\tilde{\boldsymbol{\theta}}_{c,m} | \boldsymbol{\mu}_{\tilde{\boldsymbol{\theta}}_{c,m}}^*, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}_{c,m}}^*). \tag{5.55}$$

It has been proven in [153, 159] that many of the γ_{jm}'s converge to 0 such that the corresponding feature function φ_{jm} is effectively pruned and sparse estimates for $\tilde{\boldsymbol{\theta}}_c$ are obtained. A summary of the presented training procedure is given in Algorithm 5.

Shared (hyper)parameters A slight modification of the model described above can be beneficial in cases where all latent space components $\lambda_{c,m}$ encode identical physical quantities, e.g., permeabilities in distinct subregions Ω_m. In such a case, it is reasonable to assume that the same feature functions $\varphi_{jm}(\boldsymbol{\lambda}_f)$ are activated in all

Algorithm 5 : The RVM-based prior model training algorithm, see [104].

```
**Input :** $\boldsymbol{\tau}_{cf}^{(0)}, \boldsymbol{\tau}_{c}^{(0)}, \boldsymbol{\mu}_{\tilde{\boldsymbol{\theta}}_{c,m}}^{(0)}, \boldsymbol{\gamma}^{(0)}$ ;                // Initialization
1  Evaluate all feature functions $\boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f^{(n)})$;
2  $t \leftarrow 0$
3  while (not converged) do
4      E-step: // Completely parallelizable in $N$
5      for $n = 1$ to $N$ do
6          Update $q_{\boldsymbol{\lambda}_c^{(n)}}^{(t+1)}$ via Equation (5.27)
7          Estimate $\langle \lambda_{c,m}^{(n)} \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}}$, $\langle (\lambda_{c,m}^{(n)})^2 \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}}$, $\langle u_{c,k}(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}}$, $\langle u_{c,k}^2(\boldsymbol{\lambda}_c^{(n)}) \rangle_{q_{\boldsymbol{\lambda}_c^{(n)}}^{(t)}}$
            with a suitable approximate inference method
8      end
9      M-step:
10     Find $\boldsymbol{\tau}_{cf}^{(t+1)}, \boldsymbol{\tau}_{c}^{(t+1)}, \boldsymbol{\mu}_{\tilde{\boldsymbol{\theta}}_{c,m}}^{(t+1)}$ using the update equations (5.40)–(5.42)
11     Find $\boldsymbol{\gamma}^{(t+1)}$ using the update equations (5.53), (5.54)
12     $t \leftarrow t + 1$;
13 end
14 return $\boldsymbol{\tau}_{cf}^{*}, \boldsymbol{\tau}_{c}^{*}, p(\tilde{\boldsymbol{\theta}}_c|\mathcal{D}) = \mathcal{N}(\tilde{\boldsymbol{\theta}}_{c,m}|\boldsymbol{\mu}_{\tilde{\boldsymbol{\theta}}_{c,m}}^{*}, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}_{c,m}}^{*})$
```

subregions/cells m. This can be achieved by assuming that hyperparameters γ_{jm} are identical across cells m, i.e., $\gamma_{jm} = \gamma_j$.

In such a model, the only modification concerns the update equation Equation (5.54) which becomes

$$\gamma_{jm}^{(t+1)} = \gamma_j^{(t+1)} = \frac{1}{\dim(\boldsymbol{\lambda}_c)} \sum_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \left\langle \tilde{\theta}_{c,jm}^2 \right\rangle_{q_{\tilde{\boldsymbol{\theta}}_c}^{(t)}}. \tag{5.56}$$

It is pointed out that hyperparameters γ should only be shared over equivalent physical quantities – e.g., sharing hyperparameters over components $\lambda_{c,m}$ that encode diagonal and off-diagonal entries of an anisotropic permeability tensor would not be meaningful.

An even more restrictive model is given by postulating that $\tilde{\theta}_{c,jm} = \tilde{\theta}_{c,j}$, which means that feature function coefficients are identical in all subregions Ω_m. In this case, the gradient w.r.t. $\tilde{\boldsymbol{\theta}}_c$ given by Equation (5.31) as well as the update equation for $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}_c}$ given by Equation (5.53) imply summation over m.

5.3.4 The variational relevance vector machine prior model

In our recent work published in [87], we advocate a prior model which is based on the *variational relevance vector machine* (VRVM) [161, 162], again adjusted to the unobserved target variables $\boldsymbol{\lambda}_c^{(n)}$. Just as the RVM, the VRVM does not require any hyperparameters that need to be specified by the user. The crucial advantage of the VRVM over the RVM discussed in the previous subsection is that it is fully Bayesian, i.e., it puts priors also on the remaining model parameters $\boldsymbol{\tau}_{cf}, \boldsymbol{\tau}_c$ which allows to quantify and propagate uncertainty related to those parameters. Moreover, it provides a closed-form estimate for the model evidence (or better, a rigorous lower bound) which can be used to supervise training convergence and, more importantly, allows to directly compare different model architectures, e.g., different sets/numbers of feature functions or different CGM spatial discretization resolutions.

For the parameters $\tilde{\boldsymbol{\theta}}_c$, we again use the automatic relevance determination (ARD) prior as specified in Equation (5.44),

$$p(\tilde{\boldsymbol{\theta}}_c, |\gamma) = \prod_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \prod_{j=1}^{N_{\text{features}, m}} \mathcal{N}(\tilde{\theta}_{c,jm}|0, \gamma_{jm}),$$

but instead of finding the γ_{jm}'s by maximization of the marginal likelihood $\mathcal{P}(\gamma)$, we employ a conjugate *Gamma* hyperprior on the precisions $\tau_{p,jm} = \gamma_{jm}^{-1}$ of the form

$$p(\tau_{p,jm}) = Gamma(\tau_{p,jm}|a, b) = b^a \tau_{p,jm}^{a-1} e^{-b\tau_{p,jm}} / \Gamma(a) \tag{5.57}$$

which is considered uninformative for values $a, b \ll 1$. Additionally, similar uninformative[7] conjugate *Gamma* hyperpriors are employed on the encoder and decoder precisions $\boldsymbol{\tau}_c$ and $\boldsymbol{\tau}_{cf}$,

$$p(\tau_{c,m}) = Gamma(\tau_{c,m}|c, d), \tag{5.58}$$

$$p(\tau_{cf,i}) = Gamma(\tau_{cf,i}|e, f). \tag{5.59}$$

For efficiency and model complexity reasons, the coarse-to-fine mapping $\boldsymbol{W}$ is again assumed to be the CGM shape function interpolant, $W_{ij} = \psi_j(\boldsymbol{x}_i)$.

Model training

The likelihood function is again given by Equation (5.25) such that, given the aforementioned priors, the posterior over all observable model parameters is given by

$$\begin{aligned} p(\tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c, \boldsymbol{\tau}_{cf}|\mathcal{D}) \propto \prod_{n=1}^{N} \left[\int p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\tau}_{cf}) \prod_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \left[p_c(\lambda_{c,m}^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tau_{c,m}) \right] d\boldsymbol{\lambda}_c^{(n)} \right] \\ \cdot \int p(\tilde{\boldsymbol{\theta}}_c|\boldsymbol{\tau}_p) p(\boldsymbol{\tau}_p|a, b) d\boldsymbol{\tau}_p p(\boldsymbol{\tau}_c|c, d) p(\boldsymbol{\tau}_{cf}|e, f) \end{aligned} \tag{5.60}$$

and the corresponding Bayesian graphical network is depicted in Figure 5.9. The basic concept of the VRVM is to employ a variational mean field approximation (see Section 4.3.4) on the posterior distribution over all observable and latent model

[7] In the numerical experiments of Chapter 6, we use $a = b = c = d = e = f = 10^{-10}$, which is considered to be uninformative but avoids division by 0 in the updating/model training process.

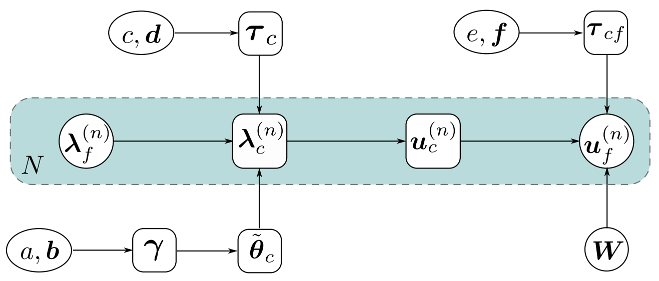


FIGURE 5.9: Graphical representation of the Bayesian network defined by the posterior given in Equation (5.60). The part with the teal background corresponds to the likelihood and is repeated N-times. Picture taken from [87].

parameters, i.e., an approximation with a factorial distribution $q(\boldsymbol{\theta})$ of the form

$$\begin{aligned}
&p\left(\left\{\tilde{\boldsymbol{\theta}}_{c,m}\right\}_{m=1}^{\dim(\boldsymbol{\lambda}_c)}, \boldsymbol{\tau}_c, \boldsymbol{\tau}_{cf}, \boldsymbol{\tau}_p, \left\{\boldsymbol{\lambda}_c^{(n)}\right\}_{n=1}^{N} \middle| \mathcal{D} \right) = \\
&= \frac{1}{p(\mathcal{D})} \prod_{n=1}^{N} \left[p_{cf}(\boldsymbol{u}_f^{(n)} | \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\tau}_{cf}) \prod_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \left[p_c(\lambda_{c,m}^{(n)} | \boldsymbol{\lambda}_f^{(n)}, \tau_{c,m}) \right] \right] p(\tilde{\boldsymbol{\theta}}_c|\boldsymbol{\gamma}) p(\boldsymbol{\gamma}) p(\boldsymbol{\tau}_c) \\
&\approx \prod_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \left[q_{\tau_{c,m}}(\tau_{c,m}) \prod_{j=1}^{N_{\text{features},m}} \left[q_{\tilde{\theta}_{c,jm}}(\tilde{\theta}_{c,jm}) q_{\tau_{p,jm}}(\tau_{p,jm}) \right] \right] \cdot \prod_{n=1}^{N} \left[q_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}) \right] \cdot \\
&\cdot \prod_{i=1}^{\dim(\boldsymbol{\lambda}_f)} \left[q_{\tau_{cf,i}}(\tau_{cf,i}) \right] = q(\boldsymbol{\theta}),
\end{aligned} \tag{5.61}$$

where $p(\mathcal{D}) = \int p(\boldsymbol{\theta}, \mathcal{D}) d\boldsymbol{\theta}$ denotes the model evidence and $\boldsymbol{\theta} = \left\{ \left\{\tilde{\boldsymbol{\theta}}_{c,m}\right\}_{m=1}^{\dim(\boldsymbol{\lambda}_c)}, \boldsymbol{\tau}_c, \boldsymbol{\tau}_{cf}, \boldsymbol{\tau}_p, \left\{\boldsymbol{\lambda}_c^{(n)}\right\}_{n=1}^{N} \right\}$ is the set of all parameters. The approximate distribution $q(\boldsymbol{\theta})$ is found by minimization of the Kullback-Leibler (KL) divergence of the mean field variational distribution $q(\boldsymbol{\theta})$ to the true posterior $p(\boldsymbol{\theta}|\mathcal{D})$ w.r.t. the factorial family as denoted above, i.e.,

$$q^*(\boldsymbol{\theta}) = \arg\min_q \mathrm{KL}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})) = \arg\min_q \left(-\int q(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta}|\mathcal{D})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta} \right). \tag{5.62}$$

It is noted that [277, 278]

$$\log p(\mathcal{D}) = \log \int p(\boldsymbol{\theta}, \mathcal{D}) d\boldsymbol{\theta} = \mathcal{F}(q) + \mathrm{KL}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})), \tag{5.63}$$

where $\mathcal{F}(q)$ is called the (log) *evidence lower bound* (ELBO)

$$\log p(\mathcal{D}) \geq \mathcal{F}(q) = \int q(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta} \tag{5.64}$$

because $\mathrm{KL}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})) \geq 0$. Since the left hand side of Equation (5.63) is independent of the variational approximation q, minimizing the KL divergence is equivalent to maximizing the ELBO $\mathcal{F}$. After maximization of the ELBO/minimization of the KL divergence, it can be assumed that $\mathrm{KL}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathcal{D})) \ll \mathcal{F}(q)$ such that $\mathcal{F}$ is an accurate approximation to the log evidence $p(\mathcal{D})$. After training, the ELBO $\mathcal{F}$ may therefore be used for Bayesian model comparison (see Section 4.2.3) which is what we use in Sections 5.6 and 6.2.10 to adjust the CGM spatial discretization.

Setting to zero the first order variations of the ELBO $\mathcal{F}$ yields (see Section 4.3.4)

$$q_{\boldsymbol{\theta}_k} = \frac{\exp \langle \log p(\boldsymbol{\theta}, \mathcal{D}) \rangle_{l \neq k}}{\int \exp \langle \log p(\boldsymbol{\theta}, \mathcal{D}) \rangle_{l \neq k} \, d\boldsymbol{\theta}_k} \tag{5.65}$$

where $\boldsymbol{\theta}_k$ is an arbitrary subset of all model parameters $\boldsymbol{\theta}$ (i.e., $\boldsymbol{\theta}_k$ practically denotes any of the parameters $\tilde{\theta}_{c,jm}, \tau_{c,m}, \tau_{cf,i}, \tau_{p,jm}, \boldsymbol{\lambda}_c^{(n)}$) and $\langle \cdot \rangle_{l \neq k}$ means expectation w.r.t. all $q_{\boldsymbol{\theta}_l}$'s except for $q_{\boldsymbol{\theta}_k}$. Obviously, every variational distribution $q_{\boldsymbol{\theta}_k}$ implicitly depends on all other $q_{\boldsymbol{\theta}_l}$, $l \neq k$ due to the above expectation values. It is therefore necessary to self-consistently loop and update over all $q_{\boldsymbol{\theta}_k}$'s using Equation (5.65) until convergence is attained.

Since the encoder and decoder p_c, p_{cf} are chosen to be linear models with Gaussian noise and we are using conjugate *Gamma* priors on the noise precisions $\boldsymbol{\tau}_c, \boldsymbol{\tau}_{cf}$ as well as on the $\tilde{\boldsymbol{\theta}}_c$-prior precision $\boldsymbol{\tau}_p$, many of the update equations following from Equation (5.65) are given in closed form,

$$\begin{aligned}
q_{\tau_{p,jm}}(\tau_{p,jm}|\tilde{a}, \tilde{b}_{jm}) &= Gamma(\tau_{p,jm}|\tilde{a}, \tilde{b}_{jm}), \\
\tilde{a} = a + \frac{1}{2}, &\qquad \tilde{b}_{jm} = b + \frac{1}{2}\left\langle \tilde{\theta}_{c,jm}^2 \right\rangle,
\end{aligned} \tag{5.66}$$

$$\begin{aligned}
q_{\tau_{c,m}}(\tau_{c,m}|\tilde{c}, \tilde{d}_m) &= Gamma(\tau_{c,m}|\tilde{c}, \tilde{d}_m), \\
\tilde{c} = c + \frac{N}{2}, &\qquad \tilde{d}_m = d + \frac{1}{2}\sum_{n=1}^{N}\left\langle \left(\lambda_{c,m}^{(n)} - \tilde{\boldsymbol{\theta}}_{c,m}^T \boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f^{(n)})\right)^2 \right\rangle,
\end{aligned} \tag{5.67}$$

$$\begin{aligned}
q_{\tau_{cf,i}}(\tau_{cf,i}|\tilde{e}, \tilde{f}_i) &= Gamma(\tau_{cf,i}|\tilde{e}, \tilde{f}_i), \\
\tilde{e} = e + \frac{N}{2}, &\qquad \tilde{f}_i = f + \frac{1}{2}\sum_{n=1}^{N}\left\langle \left[\boldsymbol{u}_f^{(n)} - \boldsymbol{W}\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)})\right]_i^2 \right\rangle,
\end{aligned} \tag{5.68}$$

$$\begin{aligned}
q_{\tilde{\theta}_{c,jm}}(\tilde{\theta}_{c,jm}) &= \mathcal{N}(\tilde{\theta}_{c,jm}|\mu_{\tilde{\theta}_{c,jm}}, \sigma_{\tilde{\theta}_{c,jm}}^2), \\
\sigma_{\tilde{\theta}_{c,jm}}^2 &= \left(\langle \tau_{c,m} \rangle \sum_{n=1}^{N} (\varphi_{jm}^{(n)})^2 + \langle \tau_{p,jm} \rangle \right)^{-1}, \\
\mu_{\tilde{\theta}_{c,jm}} &= \sigma_{\tilde{\theta}_{c,jm}}^2 \langle \tau_{c,m} \rangle \sum_{n=1}^{N} \varphi_{jm}^{(n)} \left(\left\langle \lambda_{c,m}^{(n)} \right\rangle - \sum_{k \neq j} \varphi_{km}^{(n)} \langle \tilde{\theta}_{c,km} \rangle \right),
\end{aligned} \tag{5.69}$$

where $\langle \cdot \rangle$ denotes expectation w.r.t. full $q(\boldsymbol{\theta})$ as defined by Equation (5.61). Additionally, many of the above expected values are given in closed form,

$$\begin{gathered}
\langle \tilde{\theta}_{c,jm} \rangle = \mu_{\tilde{\theta}_{c,jm}}, \qquad \left\langle \tilde{\theta}_{c,jm}^2 \right\rangle = \mu_{\tilde{\theta}_{c,jm}}^2 + \sigma_{\tilde{\theta}_{c,jm}}^2, \\
\langle \tau_{p,jm} \rangle = \frac{\tilde{a}}{\tilde{b}_{jm}}, \qquad \langle \tau_{c,m} \rangle = \frac{\tilde{c}}{\tilde{d}_m}, \qquad \langle \tau_{cf,i} \rangle = \frac{\tilde{e}}{\tilde{f}_i}.
\end{gathered} \tag{5.70}$$

The only expected values and variational distributions that are not given in closed form are the ones corresponding to the latent variables $\boldsymbol{\lambda}_c^{(n)}$ since they involve the CGM simulator $\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)})$. According to Equation (5.65), $q_{\boldsymbol{\lambda}_c^{(n)}}$ is given by

$$q_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}) \propto \left\langle p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\tau}_{cf}) \prod_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \left[p_c(\lambda_{c,m}^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \boldsymbol{\tau}_{c,m}) \right] \right\rangle \tag{5.71}$$

and the required expected values are $\left\langle \lambda_{c,m}^{(n)} \right\rangle$, $\left\langle (\lambda_{c,m}^{(n)})^2 \right\rangle$, $\left\langle \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}) \right\rangle$, $\left\langle \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}) \boldsymbol{u}_c^T(\boldsymbol{\lambda}_c^{(n)}) \right\rangle$. In principle, any of the approximate inference methods presented in Section 4.3 can be applied to estimate the above expectations. For consistency, we keep the variational nature of the model and assume variational distributions of the form

$$\begin{aligned} \tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}^2_{\boldsymbol{\lambda}_c^{(n)}}) &= \prod_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \tilde{q}_{\lambda_{c,m}^{(n)}}(\boldsymbol{\lambda}_c^{(n)}|\mu_{\lambda_{c,m}^{(n)}}, \sigma^2_{\lambda_{c,m}^{(n)}}) \\ &= \prod_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \mathcal{N}(\lambda_{c,m}^{(n)}|\mu_{\lambda_{c,m}^{(n)}}, \sigma^2_{\lambda_{c,m}^{(n)}}), \end{aligned} \tag{5.72}$$

where the variational parameters $\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}^2_{\boldsymbol{\lambda}_c^{(n)}}$ are found via black-box stochastic variational inference [211, 212], i.e., by minimization of the KL divergence

$$\boldsymbol{\mu}^*_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}^*_{\boldsymbol{\lambda}_c^{(n)}} = \arg \min_{\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}_c^{(n)}}} \mathrm{KL}\left(\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}^2_{\boldsymbol{\lambda}_c^{(n)}}) \middle|\middle| q_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}) \right) \tag{5.73}$$

between the diagonal Gaussian approximation $\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}$ and the true variational mean field distribution $q_{\boldsymbol{\lambda}_c^{(n)}}$ determined by Equation (5.71).

To solve the above optimization problem, it is noted that

$$\begin{aligned} \boldsymbol{\mu}^*_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}^*_{\boldsymbol{\lambda}_c^{(n)}} &= \arg \min_{\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}_c^{(n)}}} \mathrm{KL}\left(\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}_c^{(n)}}) \middle|\middle| q_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}) \right) \\ &= \arg \min_{\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}_c^{(n)}}} \int \tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}_c^{(n)}}) \log \frac{\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}_c^{(n)}})}{q_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)})} d\boldsymbol{\lambda}_c^{(n)} \\ &= \arg \max_{\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}_c^{(n)}}} \Bigg(\left\langle \log p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\tau}_{cf}]^{-1} \right\rangle_{\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}} \\ &\quad + \sum_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \left\langle \log p_c(\lambda_{c,m}^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tilde{\boldsymbol{\theta}}_{c,m}, \tau_{c,m}) \right\rangle_{\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}} + H\left(\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}) \right) \Bigg) \end{aligned} \tag{5.74}$$

with $H\left(\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}) \right)$ denoting the Shannon entropy of $\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)})$. Gradients w.r.t. the variational parameters $\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}_c^{(n)}}$ may readily be computed but contain expected values that involve the CGM solver $\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)})$, which can only be estimated with

Algorithm 6 : The VRVM-based prior model training algorithm, see [87].

```
Input : $\tilde{b}_{jm}^{(0)}$, $\tilde{d}_m^{(0)}$, $\tilde{f}_i^{(0)}$, $\mu_{\tilde{\theta}_{c,jm}}^{(0)}$, $(\sigma_{\tilde{\theta}_{c,jm}}^{(0)})^2$ ;                // Initialization
1  Evaluate all feature functions $\boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f^{(n)})$;
2  $\tilde{a} \leftarrow a + \frac{1}{2}$,    $\tilde{c} = c + \frac{N}{2}$,    $\tilde{e} = e + \frac{N}{2}$;
3  while (not converged) do
4      for $n \leftarrow 0$ to $N$ do
           // Fully parallelizable in $n$
5          Update $\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\mu}_{\boldsymbol{\lambda}_c}^{(n)}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}_c}^{(n)})$ using equations (5.71)–(5.78)
6          Compute $\left\langle \lambda_{c,m}^{(n)} \right\rangle_{\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}}$, $\left\langle (\lambda_{c,m}^{(n)})^2 \right\rangle_{\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}}$,
7          Estimate $\left\langle u_{c,k}(\boldsymbol{\lambda}_c^{(n)}) \right\rangle_{\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}}$, $\left\langle u_{c,k}^2(\boldsymbol{\lambda}_c^{(n)}) \right\rangle_{\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}}$ via direct Monte Carlo
8      end
9      Update $q_{\tilde{\boldsymbol{\theta}}_{c,m}}(\tilde{\boldsymbol{\theta}}_{c,m}|\boldsymbol{\mu}_{\tilde{\boldsymbol{\theta}}_{c,m}}, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}_{c,m}})$ according to (5.69) and (5.70);
10     Update $q_{\gamma_{jm}}(\gamma_{jm}|\tilde{a}, \tilde{b}_{jm})$ according to (5.66) and (5.70);
11     Update $q_{\tau_{c,m}}(\tau_{c,m}|\tilde{c}, \tilde{d}_m)$ according to (5.67) and (5.70);
12     Update $q_{\tau_{cf,i}}(\tau_{cf,i}|\tilde{e}, \tilde{f}_i)$ according to (5.68) and (5.70);
13 end
14 return Variational approximation $q(\boldsymbol{\theta})$ to $p(\boldsymbol{\theta}|\mathcal{D})$
```

Monte Carlo. The MC noise in these estimates can greatly be reduced by application of the reparametrization trick [100] (see also Section 4.3.4). The corresponding reparametrization for the diagonal Gaussian applied here takes the form

$$\lambda_{c,m}^{(n)} = \mu_{\lambda_{c,m}^{(n)}} + \sigma_{\lambda_{c,m}^{(n)}} \epsilon_m^{(n)}, \qquad \epsilon_m^{(n)} \sim \mathcal{N}(0,1). \tag{5.75}$$

Using the fact that $H\left(\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)})\right) \propto \sum_{m=1}^{\dim(\lambda_c)} \log \sigma_{\lambda_{c,m}^{(n)}}$, Equation (5.74) becomes

$$\begin{aligned}\boldsymbol{\mu}^*_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}^*_{\boldsymbol{\lambda}_c^{(n)}} = \arg \max_{\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}_{\boldsymbol{\lambda}_c^{(n)}}} \Bigg(&\left\langle \log p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}} + \boldsymbol{\sigma}_{\boldsymbol{\lambda}_c^{(n)}} \circ \boldsymbol{\epsilon}^{(n)}), \boldsymbol{\tau}_{cf}) \right\rangle_{\mathcal{N}(\boldsymbol{\epsilon}^{(n)}|\boldsymbol{0},\boldsymbol{I})} \\ + \sum_{m=1}^{\dim(\lambda_c)} \left\langle \log p_c(\mu_{\lambda_{c,m}^{(n)}} + \sigma_{\lambda_{c,m}^{(n)}} \epsilon_m^{(n)} | \boldsymbol{\lambda}_f, \tilde{\boldsymbol{\theta}}_{c,m} \tau_{c,m}) \right\rangle_{\mathcal{N}(\boldsymbol{\epsilon}^{(n)}|\boldsymbol{0},\boldsymbol{I})} &+ \sum_{m=1}^{\dim(\lambda_c)} \log \sigma_{\lambda_{c,m}^{(n)}} \Bigg),\end{aligned} \tag{5.76}$$

where we use "$\circ$" to denote element-wise multiplication. Making use of the chain rule, gradients of the above maximization problem are given by

$$\frac{\partial}{\partial \mu_{\lambda_{c,i}^{(n)}}}: \quad \left\langle \frac{\partial \lambda_{c,m}^{(n)}}{\partial \mu_{\lambda_{c,i}}^{(n)}} \frac{\partial u_{c,k}}{\partial \lambda_{c,m}^{(n)}} \frac{\partial}{\partial u_{c,k}} \log p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\tau}_{cf}) \right\rangle_{\mathcal{N}(\boldsymbol{\epsilon}^{(n)}|\boldsymbol{0},\boldsymbol{I})} + \left\langle \frac{\partial \lambda_{c,m}^{(n)}}{\partial \mu_{\lambda_{c,i}}^{(n)}} \frac{\partial}{\partial \lambda_{c,m}^{(n)}} \log p_c(\lambda_{c,m}^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tilde{\boldsymbol{\theta}}_{c,m}\tau_{c,m}) \right\rangle_{\mathcal{N}(\boldsymbol{\epsilon}^{(n)}|\boldsymbol{0},\boldsymbol{I})}, \tag{5.77}$$

$$\frac{\partial}{\partial \sigma_{\lambda_{c,i}}^{(n)}}: \quad \left\langle \frac{\partial \lambda_{c,m}^{(n)}}{\partial \sigma_{\lambda_{c,i}}^{(n)}} \frac{\partial u_{c,k}}{\partial \lambda_{c,m}^{(n)}} \frac{\partial}{\partial u_{c,k}} \log p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\tau}_{cf}) \right\rangle_{\mathcal{N}(\boldsymbol{\epsilon}^{(n)}|\boldsymbol{0},\boldsymbol{I})} + \left\langle \frac{\partial \lambda_{c,m}^{(n)}}{\partial \sigma_{\lambda_{c,i}}^{(n)}} \frac{\partial}{\partial \lambda_{c,m}^{(n)}} \log p_c(\lambda_{c,m}^{(n)}|\boldsymbol{\lambda}_f^{(n)}, \tilde{\boldsymbol{\theta}}_{c,m}\tau_{c,m}) \right\rangle_{\mathcal{N}(\boldsymbol{\epsilon}^{(n)}|\boldsymbol{0},\boldsymbol{I})} + (\sigma_{\lambda_{c,i}}^{(n)})^{-1}, \tag{5.78}$$

where repeated indices are summed and $\frac{\partial \lambda_{c,m}^{(n)}}{\partial \mu_{\lambda_{c,i}}^{(n)}} = \delta_{mi}$, $\frac{\partial \lambda_{c,m}^{(n)}}{\partial \sigma_{\lambda_{c,i}}^{(n)}} = \epsilon_m^{(n)}$ if $i = m$, and 0 else. The above gradients may then be supplied to any of the stochastic optimizers discussed in Section 4.4. Fastest convergence is observed using the ADAM optimizer [220].

It is noted that the above stochastic optimization needs to be carried out separately for every data point n, but is fully parallelizable in n. Having found the approximate variational Gaussian $\tilde{q}_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{\mu}_{\boldsymbol{\lambda}_c^{(n)}}, \boldsymbol{\sigma}^2_{\boldsymbol{\lambda}_c^{(n)}})$, the expected values $\left\langle \lambda_{c,m}^{(n)} \right\rangle$, $\left\langle (\lambda_{c,m}^{(n)})^2 \right\rangle$ are trivially given by corresponding variational mean and variance, whereas the expected values $\left\langle \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}) \right\rangle$, $\left\langle \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}) \boldsymbol{u}_c^T(\boldsymbol{\lambda}_c^{(n)}) \right\rangle$ can be estimated quickly using direct Monte Carlo. Finally, the posterior $p(\tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c, \boldsymbol{\tau}_{cf}|\mathcal{D})$ can be approximated as $p(\tilde{\boldsymbol{\theta}}_c, \boldsymbol{\tau}_c, \boldsymbol{\tau}_{cf}|\mathcal{D}) \approx q_{\tilde{\boldsymbol{\theta}}_c}(\tilde{\boldsymbol{\theta}}_c) q_{\boldsymbol{\tau}_c}(\boldsymbol{\tau}_c) q_{\boldsymbol{\tau}_{cf}}(\boldsymbol{\tau}_{cf})$. The full training algorithm is summarized in Algorithm 6.

Shared hyperparameters $\boldsymbol{\tau}_p$ As in the RVM prior model discussed in the previous subsection, the hyperparameters $\tau_{p,jm} = \gamma_{jm}^{-1}$ may be shared among latent space components $\lambda_{c,m}$, i.e., $\tau_{p,jm} = \tau_{p,j}$. In that case, the update equation (5.66) changes to

$$\tilde{b}_{jm} = \tilde{b}_j = b + \frac{1}{2} \sum_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \left\langle \tilde{\theta}_{c,jm}^2 \right\rangle \tag{5.79}$$

and thus also $\langle \tau_{p,jm} \rangle = \langle \tau_{p,j} \rangle = \frac{\tilde{a}}{\tilde{b}_j}$.

5.4 Model predictions

Irrespective of the chosen prior model, a crucial advantage of the proposed framework is that it is capable to propagate the uncertainty both due to limited model complexity, in particular the information loss during the dimension reduction process going from $\boldsymbol{\lambda}_f$ to $\boldsymbol{\lambda}_c$, and finite training data $N \lesssim 100$.

Algorithm 7 : Generation of predictive samples.

```
Input : $\boldsymbol{\lambda}_f, p(\boldsymbol{\theta}|\mathcal{D})$ ; // Test microstr. $\boldsymbol{\lambda}_f$, (approx.) posterior $p(\boldsymbol{\theta}|\mathcal{D})$
1 Evaluate all feature functions $\boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f)$;
2 Sample $\boldsymbol{\theta}_{cf}^{(s)}, \boldsymbol{\theta}_c^{(s)} \sim p(\boldsymbol{\theta}_{cf}, \boldsymbol{\theta}_c|\mathcal{D})$;
3 Sample $\boldsymbol{\lambda}_c^{(s)} \sim p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c)$;
4 Solve CGM $\boldsymbol{u}_c^{(s)} = \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(s)})$;
5 Draw predictive sample $\boldsymbol{u}_f^{(s)} \sim p_{cf}(\boldsymbol{u}_f|\boldsymbol{u}_c^{(s)}, \boldsymbol{\theta}_{cf}^{(s)})$;
6 return Predictive sample $\boldsymbol{u}_f^{(s)} \sim p_{pred}(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \mathcal{D})$ ;        // Equation (5.80)
```

Given the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ (where $\boldsymbol{\theta}$ is the set of all model parameters) after model training, the predictive distribution $p_{\text{pred}}(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \mathcal{D})$ for a new $\boldsymbol{\lambda}_f$ not contained in the training data is

$$\begin{aligned} p_{\text{pred}}(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \mathcal{D}) &= \int \underbrace{p(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \boldsymbol{\theta})}_{\text{Equation (5.3)}} p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \\ &= \int p_{cf}(\boldsymbol{u}_f|\boldsymbol{u}_c(\boldsymbol{\lambda}_c), \boldsymbol{\theta}_{cf}) p(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c) d\boldsymbol{\lambda}_c \, p(\boldsymbol{\theta}_{cf}, \boldsymbol{\theta}_c|\mathcal{D}) d\boldsymbol{\theta}_{cf} d\boldsymbol{\theta}_c \end{aligned} \tag{5.80}$$

with the encoder and decoder distributions p_c, p_{cf} as discussed in Sections 5.2.1 and 5.2.2. The integrals in Equation (5.80) are generally not analytically tractable. However, predictive samples $\boldsymbol{u}_f^{(s)} \sim p_{\text{pred}}(\boldsymbol{u}_f^{(s)}|\boldsymbol{\lambda}_f, \mathcal{D})$ can be generated inexpensively by

- drawing a parameter sample $\boldsymbol{\theta} = \{\boldsymbol{\theta}_c, \boldsymbol{\theta}_{cf}\}$ from the posterior, $\boldsymbol{\theta}_c^{(s)}, \boldsymbol{\theta}_{cf}^{(s)} \sim p(\boldsymbol{\theta}_c, \boldsymbol{\theta}_{cf}|\mathcal{D})$;
- drawing a sample for the latent space representation $\boldsymbol{\lambda}_c$ from the encoder distribution p_c, $\boldsymbol{\lambda}_c^{(s)} \sim p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c^{(s)})$;
- solving the CGM to obtain $\boldsymbol{u}_c^{(s)} = \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(s)})$; and
- drawing a predictive sample $\boldsymbol{u}_f^{(s)} \sim p_{cf}(\boldsymbol{u}_f|\boldsymbol{u}_c^{(s)}, \boldsymbol{\theta}_{cf}^{(s)})$

which only involves a forward evaluation of the much cheaper CGM instead of the FGM $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$. The above procedure can be applied to any of the prior models presented in Section 5.3[8].

It is noted though that, depending on the particular form of the (approximate) posterior $p(\boldsymbol{\theta}|\mathcal{D})$ and the encoder/decoder distributions p_c and p_{cf}, some of the integrations for the computation of lower-order statistics can be evaluated in closed form. In particular, under the assumption of Gaussian noise in the decoder p_{cf} and that the projection matrix $\boldsymbol{W}$ is fixed to the CGM shape function interpolation $W_{ij} = \psi_j(\boldsymbol{x}_i)$,

[8]Assuming $\boldsymbol{\theta}' \sim \delta(\boldsymbol{\theta}' - \boldsymbol{\theta}'_{\text{MAP}})$ for every MAP-estimated parameter subset $\boldsymbol{\theta}' \subset \boldsymbol{\theta}$.

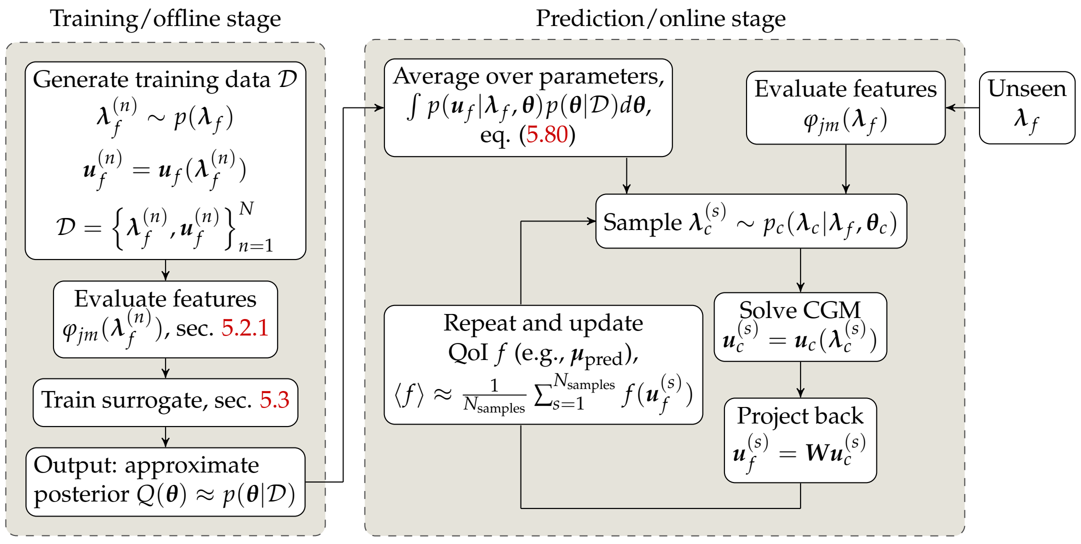

Training/offline stage
Prediction/online stage
Generate training data $\mathcal{D}$
$\boldsymbol{\lambda}_f^{(n)} \sim p(\boldsymbol{\lambda}_f)$
$\boldsymbol{u}_f^{(n)} = \boldsymbol{u}_f(\boldsymbol{\lambda}_f^{(n)})$
$\mathcal{D} = \left\{\boldsymbol{\lambda}_f^{(n)}, \boldsymbol{u}_f^{(n)}\right\}_{n=1}^N$
Evaluate features $\varphi_{jm}(\boldsymbol{\lambda}_f^{(n)})$, sec. 5.2.1
Train surrogate, sec. 5.3
Output: approximate posterior $Q(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D})$
Average over parameters, $\int p(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}$, eq. (5.80)
Evaluate features $\varphi_{jm}(\boldsymbol{\lambda}_f)$
Unseen $\boldsymbol{\lambda}_f$
Sample $\boldsymbol{\lambda}_c^{(s)} \sim p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c)$
Repeat and update QoI f (e.g., $\boldsymbol{\mu}_{\text{pred}}$), $\langle f \rangle \approx \frac{1}{N_{\text{samples}}}\sum_{s=1}^{N_{\text{samples}}} f(\boldsymbol{u}_f^{(s)})$
Solve CGM $\boldsymbol{u}_c^{(s)} = \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(s)})$
Project back $\boldsymbol{u}_f^{(s)} = \boldsymbol{W}\boldsymbol{u}_c^{(s)}$

FIGURE 5.10: Full model workflow from data generation/model training stage (left block) to prediction phase and estimation of quantities of interest (QoI) $f(\boldsymbol{u}_f)$. Picture adapted from [87, 104].

the predictive mean $\boldsymbol{\mu}_{\text{pred}}(\boldsymbol{\lambda}_f)$ is

$$\begin{aligned}\boldsymbol{\mu}_{\text{pred}}(\boldsymbol{\lambda}_f) &= \int \boldsymbol{u}_f p_{\text{pred}}(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \mathcal{D})d\boldsymbol{u}_f \\ &\approx \frac{1}{N_{\text{samples}}}\boldsymbol{W}\sum_{s=1}^{N_{\text{samples}}} \boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(s)}),\end{aligned} \tag{5.81}$$

where the samples $\boldsymbol{\lambda}_c^{(s)}$ are generated according to the first two points of the above list. Similarly, the predictive variance $\boldsymbol{\sigma}^2_{\text{pred}}(\boldsymbol{\lambda}_f)$ is given by

$$\begin{aligned}\boldsymbol{\sigma}^2_{\text{pred}}(\boldsymbol{\lambda}_f) &= \int \boldsymbol{u}_f^2 p_{\text{pred}}(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \mathcal{D})d\boldsymbol{u}_f - \left(\int \boldsymbol{u}_f p_{\text{pred}}(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \mathcal{D})d\boldsymbol{u}_f\right)^2 \\ &= \int \left((\boldsymbol{W}\boldsymbol{u}_c(\boldsymbol{\lambda}_c))^2 + \boldsymbol{\tau}_{cf}^{-1}\right) p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c)p(\boldsymbol{\tau}_{cf}, \boldsymbol{\theta}_c|\mathcal{D})d\boldsymbol{\tau}_{cf}d\boldsymbol{\theta}_c - \boldsymbol{\mu}^2_{\text{pred}} \\ &= \frac{1}{N_{\text{samples}}}\sum_{s=1}^{N_{\text{samples}}} (\boldsymbol{W}\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(s)}))^2 - \boldsymbol{\mu}^2_{\text{pred}} + \left\langle \boldsymbol{\tau}_{cf}^{-1}\right\rangle_{p(\boldsymbol{\tau}_{cf}, \boldsymbol{\theta}_c|\mathcal{D})},\end{aligned} \tag{5.82}$$

where the square/inverse of a vector denotes element-wise square/inverse and the samples $\boldsymbol{\lambda}_c^{(s)}$ are generated as described above. For the models assuming $p(\boldsymbol{\tau}_{cf}|\mathcal{D}) = \delta(\boldsymbol{\tau}_{cf} - \boldsymbol{\tau}_{cf,\text{MAP}})$ (i.e., the maximum-likelihood approach Section 5.3.1, the Laplacian prior Section 5.3.2 and the RVM-based prior Section 5.3.3), the expected value of the decoder precision $\boldsymbol{\tau}_{cf}$ is identical to its MAP estimate, $\left\langle \boldsymbol{\tau}_{cf}^{-1}\right\rangle_{p(\boldsymbol{\tau}_{cf}, \boldsymbol{\theta}_c|\mathcal{D})} = \boldsymbol{\tau}_{cf,\text{MAP}}$. Assuming the approximate posterior $p(\tau_{cf,i}|\mathcal{D}) = q_{\tau_{cf,i}}(\tau_{cf,i}) = Gamma(\tau_{cf,i}|\tilde{e}, \tilde{f}_i)$ as given by Equation (5.68) in the VRVM-based prior model Section 5.3.4, the expected value is given in closed form as $\left\langle \tau_{cf,i}^{-1}\right\rangle_{p(\tau_{cf,i}, \boldsymbol{\theta}_c|\mathcal{D})} = \frac{\tilde{f}_i}{\tilde{e}-1}$.

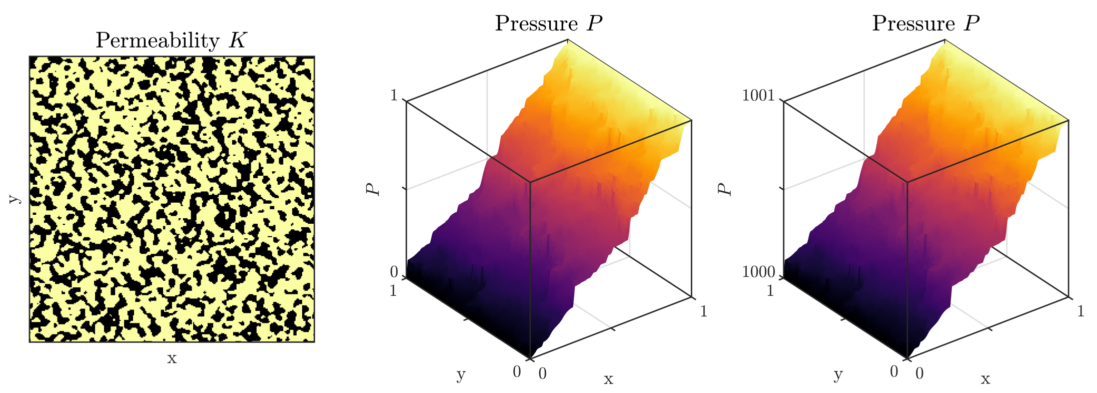


FIGURE 5.11: Binary permeability field with $K_{lo} = 1$, $K_{hi} = 100$ (left) and the corresponding Darcy flow pressure response fields P with zero flux on the top/bottom boundaries and $P^{(1)}_{\text{left}} = 0$, $P^{(1)}_{\text{right}} = 1$ (middle) and $P^{(2)}_{\text{left}} = 1000$, $P^{(2)}_{\text{right}} = 1001$ (right) essential pressure boundary conditions. The suggested surrogate model is aware of essential boundary conditions, i.e., it fits the data perfectly at the left/right side. It is clear that the error metric ϵ defined by Equation (5.83) is much lower for a dataset with boundary conditions as in the right plot compared to the middle one. The metric ϵ can therefore only be used for comparison of data with identical boundary conditions.

The algorithm for the generation of predictive samples is summarized in Algorithm 7. The full model workflow, i.e., training data generation, model training, and prediction stage is summarized graphically in the flowchart of Figure 5.10.

5.4.1 Model performance metrics

The quality of a surrogate model is determined by (a) its computational efficiency and (b) its predictive accuracy, i.e., its proximity to the true FGM solution $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$ and the correctness of predictive uncertainty measures such as $\boldsymbol{\sigma}_{\text{pred}}$ given by Equation (5.82). During the development of this work, several performance metrics have been applied to measure both proximity and the quality of predictive uncertainty.

A simple, yet very common error measure that is used in [104] is the relative L_2 distance ϵ defined by

$$\epsilon = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \frac{\|\boldsymbol{u}_f^{(n)} - \boldsymbol{\mu}_{\text{pred}}(\boldsymbol{\lambda}_f^{(n)})\|}{\|\boldsymbol{u}_f^{(n)}\|} \tag{5.83}$$

which is normalized by the L_2-norm $\|\boldsymbol{u}_f^{(n)}\|$ of every test sample $\boldsymbol{u}_f^{(n)}$. Although ϵ is a relative error measure, it does not allow for seamless comparison of datasets which have considerably different $\langle \|\boldsymbol{u}_f\| \rangle = \frac{1}{N_{\text{test}}} \sum_n \|\boldsymbol{u}_f^{(n)}\|$. Assume, for instance, two datasets of Darcy flow in porous media where on the left/right side of a unit square domain the pressure is fixed to $P_{\text{left}}, P_{\text{right}}$ and on the top/bottom sides a zero flux boundary condition is applied, see the visualization in Figure 5.11. Assume further that the first dataset has $P^{(1)}_{\text{left}} = 0, P^{(1)}_{\text{right}} = 1$ whereas the second one has

$P_{\text{left}}^{(2)} = 1000, P_{\text{right}}^{(2)} = 1001$[9]. It is obvious that a surrogate model trained on both datasets will exhibit considerably lower ϵ on the second dataset because $\langle \|\boldsymbol{u}_f\| \rangle$ is much higher but $\left\langle \|\boldsymbol{u}_f^{(n)} - \boldsymbol{\mu}_{\text{pred}}(\boldsymbol{\lambda}_f^{(n)})\| \right\rangle$ is (at least approximately) the same as for the first dataset.

Another drawback of the performance metric defined by ϵ is that it does not take into account the *variance* of the data. It is clear that it is much easier for a surrogate to produce low L_2-errors when the output distribution $p(\boldsymbol{u}_f)$ is narrow. A proximity performance metric that allows the comparison of the quality of the surrogate on various different datasets should therefore not be normalized by the L_2-norm $\|\boldsymbol{u}_f\|$, but the *variance* of the data. The output data variance can be estimated by

$$\operatorname{var}(u_{f,i}) \approx \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} (u_{f,i}^{(n)} - \bar{u}_{f,i})^2, \tag{5.84}$$

where $\bar{u}_{f,i} = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} u_{f,i}^{(n)}$ is the sample mean of the test set for component i of the FGM response vector. One such performance measure is introduced in [104] as

$$e = \frac{1}{N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \frac{1}{\dim(\boldsymbol{u}_f)} \sum_{i=1}^{\dim(\boldsymbol{u}_f)} \frac{(u_{f,i}^{(n)} - \mu_{\text{pred},i}(\boldsymbol{\lambda}_f^{(n)}))^2}{\operatorname{var}(u_{f,i})}. \tag{5.85}$$

The quantity $e \geq 0$ has the property to be $e = 1$ if the surrogate prediction is identical to the mean of the (test) data, i.e., if $\mu_{\text{pred},i}(\boldsymbol{\lambda}_f^{(n)}) = \bar{u}_{f,i}$ for all test samples n. A suitable surrogate should therefore provide values for e that are considerably smaller than 1.

A closely related, but much more popular performance metric is the *coefficient of determination* R^2 [279, 280] which is used in the work published in [87]. It is defined by

$$R^2 = 1 - \frac{\sum_{n=1}^{N_{\text{test}}} \|\boldsymbol{u}_f^{(n)} - \boldsymbol{\mu}_{\text{pred}}(\boldsymbol{\lambda}_f^{(n)})\|^2}{\sum_{n=1}^{N_{\text{test}}} \|\boldsymbol{u}_f^{(n)} - \bar{\boldsymbol{u}}_f\|^2} \tag{5.86}$$

and measures the proportion of variance of the test data that is explained by the model prediction $\boldsymbol{\mu}_{\text{pred}}(\boldsymbol{\lambda}_f^{(n)})$. Assuming again that the model prediction is as poor as the test data mean, i.e., $\boldsymbol{\mu}_{\text{pred}}(\boldsymbol{\lambda}_f^{(n)}) = \bar{\boldsymbol{u}}_f$, the coefficient of determination is $R^2 = 0$, i.e., 0% of the variance of the test data are explained by the model prediction $\boldsymbol{\mu}_{\text{pred}}$. On the other side, if the surrogate can perfectly predict the FGM output, i.e., $\boldsymbol{u}_f^{(n)} = \boldsymbol{\mu}_{\text{pred}}(\boldsymbol{\lambda}_f^{(n)})$, the coefficient of determination is $R^2 = 1$.

A drawback of the coefficient of determination R^2 is that it is incapable to assess the quality of the whole predictive distribution $p_{\text{pred}}(\boldsymbol{u}_f | \boldsymbol{\lambda}_f, \mathcal{D})$. Imagine for instance that $p_{\text{pred}}(\boldsymbol{u}_f^{(n)} | \boldsymbol{\lambda}_f^{(n)}, \mathcal{D}) = \delta(\boldsymbol{u}_f^{(n)} - \boldsymbol{\mu}_{\text{pred}}(\boldsymbol{\lambda}_f^{(n)}))$: even when $\|\boldsymbol{u}_f^{(n)} - \boldsymbol{\mu}_{\text{pred}}(\boldsymbol{\lambda}_f^{(n)})\|^2$ is very small (but nonzero) $\forall n$ and therefore $R^2 \approx 1$, the likelihood to observe $\boldsymbol{\lambda}_f^{(n)}, \boldsymbol{u}_f^{(n)}$

[9]In arbitrary units.

under the predictive distribution is in fact 0. It is therefore necessary to estimate the likelihood of the test data under the predictive distribution. This is realized in [87] by measuring the *mean log likelihood* (MLL, see also [46]) under the assumption that p_{pred} is a diagonal Gaussian $p_{\text{pred}}(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \mathcal{D}) = \mathcal{N}(\boldsymbol{u}_f|\boldsymbol{\mu}_{\text{pred}}(\boldsymbol{\lambda}_f), \text{diag}(\boldsymbol{\sigma}^2_{\text{pred}}))$,

$$\begin{aligned} MLL &= \frac{1}{\dim(\boldsymbol{u}_f) N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \log p_{\text{pred}}(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \mathcal{D}) \\ &= -\frac{\log(2\pi)}{2} - \frac{1}{2\dim(\boldsymbol{u}_f) N_{\text{test}}} \sum_{n=1}^{N_{\text{test}}} \sum_{i=1}^{\dim(\boldsymbol{u}_f)} \left(\log \sigma^2_{\text{pred},i}(\boldsymbol{\lambda}_f^{(n)}) + \frac{\left(u_{f,i}^{(n)} - \mu_{\text{pred},i}^{(n)}\right)^2}{\sigma^2_{\text{pred}}(\boldsymbol{\lambda}_f^{(n)})} \right), \end{aligned} \tag{5.87}$$

where $\mu_{\text{pred},i}^{(n)} = \mu_{\text{pred},i}(\boldsymbol{\lambda}_f^{(n)})$. It is noted that in the case of a discrete predictive distribution $p_{\text{pred}}(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \mathcal{D})$, the highest possible value is $MLL = 1$ for perfect prediction $p_{\text{pred}}(\boldsymbol{u}_f(\boldsymbol{\lambda}_f)|\boldsymbol{\lambda}_f, \mathcal{D}) = 1, p_{\text{pred}}(\boldsymbol{u}_f \neq \boldsymbol{u}_f(\boldsymbol{\lambda}_f)|\boldsymbol{\lambda}_f, \mathcal{D}) = 0$ where $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$ is the true FGM solution.

5.5 Numerical complexity analysis

In general, the workflow of any kind of surrogate model can be split up in an *offline stage* where the model is set up and calibrated (model training), and an *online stage* where the surrogate accomplishes its purpose by providing approximations to the original simulation at a desirably much lower cost (model prediction), see Figure 5.10. Clearly, in a proper numerical complexity analysis, both stages need to be distinguished and more attention has to be paid on the online (= prediction) stage as it is critical for any surrogate to be computationally more efficient than the original model/simulation that is to be replaced.

In the proposed model, the following quantities are identified to be potentially relevant for the numerical complexity of the training and/or prediction stage: The number of training data N, the FGM input and output dimensions $\dim(\boldsymbol{\lambda}_f)$ and $\dim(\boldsymbol{u}_f)$, the CGM input and output dimensions $\dim(\boldsymbol{\lambda}_c), \dim(\boldsymbol{u})c)$, and the number of feature functions per latent space variable $\dim(\tilde{\boldsymbol{\theta}}_{c,m})$[10].

As can be concluded from Algorithms 4–6, the training (offline) phase generally scales linear with the number of training samples N because of the inner for-loop. However, because of the factorization of the variational distributions $q_{\boldsymbol{\lambda}_c^{(1)},\ldots,\boldsymbol{\lambda}_c^{(N)}}(\boldsymbol{\lambda}_c^{(1)}, \ldots, \boldsymbol{\lambda}_c^{(N)}) = \prod_{n=1}^N q_{\boldsymbol{\lambda}_c^{(n)}}(\boldsymbol{\lambda}_c^{(n)})$, inference w.r.t. $q_{\boldsymbol{\lambda}_c^{(1)},\ldots,\boldsymbol{\lambda}_c^{(N)}}(\boldsymbol{\lambda}_c^{(1)}, \ldots, \boldsymbol{\lambda}_c^{(N)})$ is fully parallelizable in N. After training, the optimal model parameters/variational posterior distributions are passed to the prediction routine which is completely independent of the training data and therefore scales as $\mathcal{O}(1)$.

[10]In all experiments of Chapter 6, the same set of features is applied for every latent space component m, i.e., $\dim(\tilde{\boldsymbol{\theta}}_{c,m})$ is independent of m.

Quantity / Phase	N	$\dim(\boldsymbol{u}_f)$	$\dim(\boldsymbol{\lambda}_f)$	$\dim(\boldsymbol{u}_c)$	$\dim(\boldsymbol{\lambda}_c)$	$\dim(\tilde{\boldsymbol{\theta}}_{c,m})$
training	N	$\dim(\boldsymbol{u}_f)$	$1 \dots (\dim(\boldsymbol{\lambda}_f))^2$	$\dim(\boldsymbol{u}_c)$	$\dim(\boldsymbol{\lambda}_c)$	$\dim(\tilde{\boldsymbol{\theta}}_{c,m})$, $(\dim(\tilde{\boldsymbol{\theta}}_{c,m}))^3$
prediction	1	$\dim(\boldsymbol{u}_f)$	$1 \dots (\dim(\boldsymbol{\lambda}_f))^2$	$\dim(\boldsymbol{u}_c)$	$\dim(\boldsymbol{\lambda}_c)$	$\dim(\tilde{\boldsymbol{\theta}}_{c,m})$

TABLE 5.1: Numerical complexity of training and prediction phases. The scaling with $\dim(\boldsymbol{\lambda}_f)$ depends on the set of feature functions $\boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f)$ used and reaches from $\mathcal{O}(1)$ to $\mathcal{O}\left((\dim(\boldsymbol{\lambda}_f))^2\right)$. The scaling w.r.t. $\dim(\tilde{\boldsymbol{\theta}}_{c,m})$ is linear in the VRVM model but cubic for the Laplacian and RVM-based priors.

Assuming that the reconstruction in the decoder distribution p_{cf} corresponds to the linear projection $\boldsymbol{u}_f = \boldsymbol{W}\boldsymbol{u}_c + \boldsymbol{\tau}_{cf}^{-1/2} \circ \boldsymbol{Z}$, both model training and prediction scale linearly with $\dim(\boldsymbol{u}_f)$.

Also, $\boldsymbol{\lambda}_c$ is only involved via matrix-vector operations such that the scaling is $\mathcal{O}(\dim(\boldsymbol{\lambda}_c))$ for both stages.

The scaling with the fine scale microstructure resolution $\dim(\boldsymbol{\lambda}_f)$ is fully dependent on the applied feature functions φ_{jm} and ranges from $\mathcal{O}(1)$ for, e.g., the constant feature, and $\mathcal{O}\left((\dim(\boldsymbol{\lambda}_f))^2\right)$ for, e.g., features that involve mutual exclusion/pixel interactions[11]. It is noted that the microstructural features can be precomputed and stored to be reused, e.g., for problems under different boundary conditions.

The scaling with the CGM resolution/degrees of freedom $\dim(\boldsymbol{u}_c)$ depends on the applied constitutive laws and numerical solvers that are applied. If the CGM corresponds to Darcy flow and a sufficiently regular PDE discretization is used, a sparse banded solver [281] can be applied which scales linearly, $\mathcal{O}(\dim(\boldsymbol{u}_c))$. The worst case for a linear PDE with dense stiffness matrix is $\mathcal{O}\left((\dim(\boldsymbol{u}_c))^2\right)$ which is typically still much faster than an FGM solution because $\dim(\boldsymbol{u}_c) \ll N_{dof,f}$, where $N_{dof,f}$ are the degrees of freedom of the FGM.

The training phase scaling with the number of feature functions per latent space component $\dim(\tilde{\boldsymbol{\theta}}_m)$ depends on the model specifics. As can be seen on Equations (5.42) and (5.53), (5.54), training scales as $\mathcal{O}\left((\dim(\tilde{\boldsymbol{\theta}}_{c,m}))^3\right)$ for the Laplacian and RVM prior model which is the main reason why not more than a few hundred features are used in the experiments of Chapter 6. For the VRVM prior model though, as correlations between $\tilde{\theta}_{c,jm}$ and $\tilde{\theta}_{c,j'm}$, $j \neq j'$ are neglected, the scaling is linear, $\mathcal{O}(\dim(\tilde{\boldsymbol{\theta}}_{c,m}))$. For predictions, all proposed prior models only involve matrix-vector operations w.r.t. $\tilde{\theta}_{c,m}$ such that the scaling is linear.

[11]It is possible to include more expensive features – however, the surrogate would defeat the purpose if feature function evaluation becomes more expensive than an FGM forward run.

The scaling behavior in training and prediction stages w.r.t. all the mentioned quantities is summarized in Table 5.1.

5.6 Automated coarse-grained model refinement

As presented in the model overview of Section 5.2, the coarse-grained model (CGM) $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ is given by a numerical solver operating on substantially larger length scales and potentially simplified constitutive laws compared to the fine-grained model (FGM) $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$ that is to be replaced by the probabilistic surrogate. In particular, both CGM and FGM solvers correspond to finite element simulations, where the CGM is discretized on a considerably coarser scale than the FGM and is therefore much more quickly solvable.

Given a set of fine scale training data $\mathcal{D} = \left\{\boldsymbol{\lambda}_f^{(n)}, \boldsymbol{u}_f^{(n)}\right\}_{n=1}^{N}$, it is an open question what discretization for the CGM solver $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ is optimal with respect to computational efficiency, but also with regard to model complexity. As the latent space of $\boldsymbol{\lambda}_c$'s constitutes the model bottleneck to which information contained in the microstructure $\boldsymbol{\lambda}_f$ is squeezed, it plays a pivotal role in the model's generalization capabilities given a finite dataset $\mathcal{D}$. Moreover, it is unclear which specific shape of PDE discretization (i.e., vertex coordinates, element connectivity, etc.) performs best in terms of predictive error under a given computational budget.

To address these questions, the model evidence $p(\mathcal{D}) = \int \mathcal{L}(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$ can be considered as a quality metric of how well a model with a certain CGM discretization fits the data $\mathcal{D}$, see Section 4.2.3. Conveniently, the evidence lower bound (ELBO), which is an approximation to the model evidence $p(\mathcal{D})$, comes out as a by-product in the training phase of the VRVM model discussed in Section 5.3.4. The naive approach to find a CGM discretization which is (sub)optimal under the constraint of a fixed number of vertices/elements is to train all models of a certain class of meshes that fulfill these requirements and to pick the model with highest ELBO afterwards. This approach is fine from a theoretical point of view, but requires an exceedingly large effort for model training because the number of possible discretizations (and therefore the number of models to be trained) increases combinatorically with the number of vertices/elements that are prescribed.

For that reason, an evidence-based refinement strategy is introduced in [87] which finds a (sub)optimal effective material property random field discretization consisting of square elements of different size as depicted in Figure 5.12. It is noted that the discretization of the effective material property random field $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c)$ (see Equation (5.7)) not necessarily coincides with the PDE discretization of the CGM. Rather, the material property field discretization needs to be resolvable by the PDE discretization, i.e., within a finite element, only constant material properties, i.e., constant Darcy permeability, are allowed. Clearly, with regard to computational efficiency, it is optimal if PDE and material property field discretizations coincide. However,

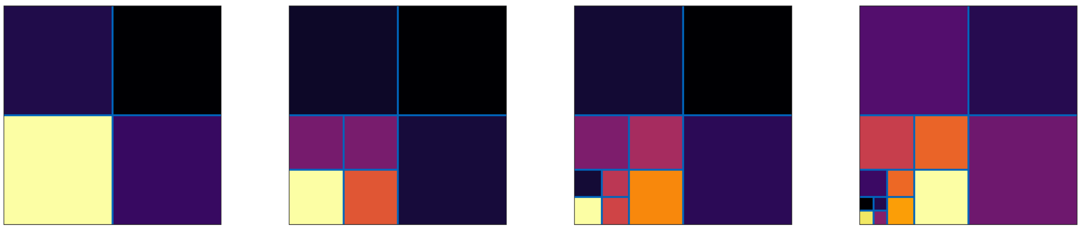


FIGURE 5.12: Schematic representation of the refinement procedure. Starting from a 2×2 square grid, a square cell is chosen according to a scoring function (color scale) and split into four square subcells. The number of cells, i.e., the latent space dimension $\dim(\boldsymbol{\lambda}_c)$, which represents the information bottleneck, increases by 3 with each split.

for pure investigation of the influence of effective material property field discretization, it is important to suppress shamming influence of PDE discretization errors. Therefore, it is advisable to compare CGMs with constant PDE- but altering material property field discretizations only.

The basic idea behind the proposed refinement strategy is to start from a 2×2 square grid[12] for the discretization of the effective material property field $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c)$ which is of the form defined by Equation (5.7), (5.8). The model is trained to full convergence and then, according to a heuristic scoring function, a subregion/cell m is chosen where refinement seems to be most promising. The chosen square subregion/cell m is then split by dividing the square into for equally sized sub-squares each of constant material property, increasing the latent space dimension (and therefore the size of the bottleneck) as $\dim(\boldsymbol{\lambda}_c) \leftarrow \dim(\boldsymbol{\lambda}_c) + 3$. Next, the model is retrained to convergence, with a parameter initialization deduced from the aforegoing optimum. This process is repeated until the previously defined number of elements is attained.

To motivate the scoring function that chooses the subregion m that is to be refined, we focus on the model ELBO. According to Equation (5.64), the ELBO in the VRVM model of Section 5.3.4 is given by

$$\mathcal{F}(q) = \left\langle \log p(\boldsymbol{\theta}, \mathcal{D}) - q(\boldsymbol{\theta}) \right\rangle_q \leq p(\mathcal{D}), \tag{5.88}$$

where the joint and variational mean-field distribution $p(\boldsymbol{\theta}, \mathcal{D})$, $q(\boldsymbol{\theta})$ are given by Equation (5.61). Since the above expected value is w.r.t. the variational distributions q, it can be evaluated explicitly and is given by

$$\begin{aligned}\mathcal{F}(q) = &- \tilde{e} \sum_{i=1}^{\dim(\boldsymbol{u}_f)} \log \tilde{f}_i + \sum_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \sum_{n=1}^{N} \log \sigma_{\lambda_c,m}^{(n)} - \tilde{c} \sum_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \log \tilde{d}_m \\ &- \tilde{a} \sum_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \sum_{j=1}^{\dim(\boldsymbol{\tau}_p)} \log \tilde{b}_{jm} + \sum_{m=1}^{\dim(\boldsymbol{\lambda}_c)} \sum_{j=1}^{\dim(\boldsymbol{\tau}_p)} \log \sigma_{\tilde{\theta}_{c,jm}}.\end{aligned} \tag{5.89}$$

[12] We always work on a $2D$ square domain.

As every latent space component $\lambda_{c,m}$ encodes the permeability tensor $\boldsymbol{K}_m = e^{\lambda_{c,m}}\boldsymbol{I}$ of exactly one subregion m, we search the component m which we think can increase $\mathcal{F}$ the most when split. Therefore, all terms that can directly be assigned to a component m are regrouped into a function $\mathcal{F}_m(q)$ which is

$$\mathcal{F}_m(q) = \sum_{n=1}^{N} \log \sigma_{\lambda_{c,m}}^{(n)} - \tilde{c} \log \tilde{d}_m - \tilde{a} \sum_{j=1}^{\dim(\boldsymbol{\tau}_p)} \log \tilde{b}_{jm} + \sum_{j=1}^{\dim(\boldsymbol{\tau}_p)} \log \sigma_{\tilde{\theta}_{c,jm}}. \tag{5.90}$$

In the case of shared feature function precisions $\tau_{p,jm} = \tau_{p,j}$, this reduces to

$$\mathcal{F}_m(q) = \sum_{n=1}^{N} \log \sigma_{\lambda_{c,m}}^{(n)} - \tilde{c} \log \tilde{d}_m + \sum_{j=1}^{\dim(\boldsymbol{\tau}_p)} \log \sigma_{\tilde{\theta}_{c,jm}}. \tag{5.91}$$

The heuristic we use consists of picking the subregion m for splitting which has least $\mathcal{F}_m(q)$ and therefore seems to have most potential to increase the ELBO $\mathcal{F}$. The above scoring function suggests that the variational latent space variances $\sigma_{\lambda_{c,m}}^{(n)}$ should be small, whereas the decoder misfit $\tilde{d}_m = d + \frac{1}{2}\sum_{n=1}^{N} \left\langle \left(\lambda_{c,m}^{(n)} - \tilde{\boldsymbol{\theta}}_{c,m}^{T} \boldsymbol{\varphi}_m(\boldsymbol{\lambda}_f^{(n)})\right)^2 \right\rangle$ (see Equation (5.67)) should be large and there should be high certainty about the feature coefficients $\tilde{\theta}_{c,jm}$, i.e., low variance $\sigma_{\tilde{\theta}_{c,jm}}$. Clearly, this is nothing more than a heuristic with no proof of an increased ELBO after splitting.

5.7 Chapter summary

The scope of this chapter is to propose a machine learning framework for surrogate modeling of expensive numerical simulations of Stokes and Darcy flow through random porous media, denoted as the *fine grained model* (FGM) $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$. An accurate, fine scale description of random media typically requires a large number of descriptors $\lambda_{f,i}$ which correspond to the high-dimensional, stochastic input to the expensive FGM simulation. The high input dimensionality together with the small number of feasible FGM evaluations seriously hampers the construction of cheap and accurate machine learning surrogates.

The proposed remedy is to construct the machine learning surrogate using a reduced order simulator $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ based on coarser spatial discretizations and possibly simplified governing equations, referred to as the coarse grained model or CGM. The CGM requires as input a much lower dimensional, homogenized encoding $\boldsymbol{\lambda}_c$ of the high dimensional microstructural description $\boldsymbol{\lambda}_f$.

This dimension reduction is carried out by the encoder distribution $p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c)$. Various choices of modeling p_c are conceivable. To optimally exploit a priori knowledge about the physics of the underlying problem (porous medium flow), we suggest a Gaussian linear model with a large library of predefined feature/basis functions $\varphi_{jm}(\boldsymbol{\lambda}_f)$ primarily inspired by the rich literature on random heterogeneous media (see [1] as a principal reference), but also by image recognition tools or autoencoder representations.

To avoid exuberant model complexity due to the large library of feature functions applied in the encoder p_c, different types of sparsity enforcing prior models are presented in Sections 5.3.2–5.3.4, together with efficient model training algorithms. Moreover, a sparse set of activated feature functions allows (a) for model interpretability and (b) to ignore pruned features in the prediction stage, which in turn enables a computationally more efficient online phase of generating predictive samples.

The decoder density $p_{cf}(\boldsymbol{u}_f|\boldsymbol{u}_c(\boldsymbol{\lambda}_c), \boldsymbol{\theta}_{cf})$ is chosen to be a linear projection $u_{f,i} = W_{ij}u_{c,j} + \tau_{cf,i}^{-1/2}Z_i$ with Gaussian noise $Z_i \sim \mathcal{N}(0, 1)$. The projection matrix $\boldsymbol{W}$ is set to the coarse model shape function interpolant, $W_{ij} = \psi_j(\boldsymbol{x}_i)$, while the precision parameter $\boldsymbol{\tau}_{cf}$ is learned from the data. Again, many different models are conceivable, e.g., Gaussian processes, (deep) neural networks, etc. However, following the precept of maximal incorporation of a priori information, fixing $W_{ij} = \psi_j(\boldsymbol{x}_i)$ is the most obvious choice.

The chapter is completed by a detailed discussion of how the predictive accuracy of the surrogate can be measured, how numerical complexity scales in training and prediction stages, and a suggestion of how the CGM can adaptively be refined to widen the information bottleneck given by the latent, low dimensional variables $\boldsymbol{\lambda}_c$.

Chapter 6

Numerical experiments

This chapter is based on the findings published in

C. Grigo, P.-S. Koutsourelakis:

"A physics-aware, probabilistic machine learning framework for coarse-graining high-dimensional systems in the Small Data regime",
Elsevier Journal of Computational Physics 2019, Volume 397, 108842

C. Grigo, P.-S. Koutsourelakis:

"Bayesian Model and Dimension Reduction for Uncertainty Propagation: Applications in Random Media",
SIAM/ASA Journal on Uncertainty Quantification 2019 7:1, 292-323

C. Grigo, P.-S. Koutsourelakis:

"Probabilistic reduced-order modeling for stochastic partial differential equations",
Eccomas Proceedia UNCECOMP (2017) 111-129

The numerical experiments performed in this work aim to answer the following questions:

- **Validation:** Is the surrogate model proposed in Chapter 5 able to fully reproduce closed form solutions/solutions in the homogenization limit?
- **Predictive performance:** In terms of the performance metrics introduced in Section 5.4.1, how well does the proposed model perform in dependence of (a) the number of training data N and (b) the coarse model resolution/latent space dimension $\dim(\boldsymbol{\lambda}_c)$?
- **Effective material properties:** Do effective permeabilities assume reasonable values?
- **Predictive uncertainty:** How accurate is the predictive uncertainty σ_{pred}?

- **Sparsity:** Is the set of activated features really sparse? Does the number of activated features change with N?
- **Interpretability:** Which feature functions are activated? Are different feature functions activated for different fine scale data distributions $\boldsymbol{\lambda}_f \sim p(\boldsymbol{\lambda}_f)$?
- **Extrapolative capabilities:** Is it possible to get accurate predictions even on test data with different boundary conditions than used in the training data?
- **Uncertainty propagation:** How well does the model perform in the use case of an uncertainty propagation (UP) problem?
- **Adaptive refinement:** Can the suggested adaptive refinement procedure outperform homogeneously discretized models of same complexity?

Most of the above questions are answered both for Stokes and Darcy FGM simulations. Others are, however, only answered for either Stokes or Darcy flow. Moreover, only some of the above questions are addressed for all three prior models suggested in Sections 5.3.2–5.3.4. This is partially because some of the above points can only be treated under a certain setting (e.g., the ELBO for model comparison required for the adaptive refinement procedure is only given in the VRVM based model introduced in Section 5.3.4). More importantly though, the prior models have been developed at different times during the genesis of this work – just as the above questions did not arise all at the same time. In all of the following experiments, we give a clear indication of the model and data specifics that are used and strive to keep this section as clear as possible.

6.1 Same physics: Darcy flow fine scale data

In this section, we exclusively use FGM data from a Darcy flow simulator, see Sections 2.2 and 3.1.1, which we denote by $\boldsymbol{u}_f(\boldsymbol{\lambda}_f) : \boldsymbol{\lambda}_f \mapsto \boldsymbol{u}_f$. The surrogate core unit, i.e., the CGM solver $\boldsymbol{u}_c(\boldsymbol{\lambda}_c) : \boldsymbol{\lambda}_c \mapsto \boldsymbol{u}_c$, is based on Darcy's equations of motion as well but solved on much coarser finite element discretizations. The results of the different experiments have been published in [103, 104].

6.1.1 Validation: a one-dimensional example

To validate the suggested algorithm, we consider a one-dimensional example where the FGM corresponds to a standard Galerkin finite element simulation of the one-dimensional PDE given in Equation (3.3) with $N_{el,f} = 128$ equally sized finite elements. A binary material of contrast 10 is used, i.e., the low and high phase permeabilities are set to $K_{lo} = 1$, $K_{hi} = 10$. The domain Ω is given by unit interval, $\Omega = [0, 1]$, where the left end $x = 0$ is an essential pressure boundary, $\Gamma_P = \{0\}$, where $P = 0$. The right end $x = 1$ is a natural boundary, $\Gamma_V = \{1\}$, where $V_{bc} = -100$ is specified. To each of the 128 finite elements, a permeability of either

K_{lo} or K_{hi} is assigned according to a level-cut Gaussian process sample

$$g(x) \sim GP(0, k(x - x')) \tag{6.1}$$

with kernel function $k(x - x') = \exp\left\{-\frac{(x-x')^2}{l^2}\right\}$, $l = 0.01$ and a binary discretization related to the expected volume fractions $|\Omega_f|, |\Omega_s| = 1 - |\Omega_f|$ as given in Equations(2.19), (2.20). The input microstructural description $\boldsymbol{\lambda}_f$ is given by a 128-dimensional binary vector containing the permeability of each element. The Gaussian process cutoff parameter c_{cut} controlling the volume fraction is chosen such that $|\Omega_f| \sim U(0, 1)$.

Model specifications

The CGM is given by a standard finite element solver with $N_{el,c} = 8$ elements, i.e., $\dim(\boldsymbol{\lambda}_c) = 8$ and 8 fine scale elements are contained in a coarse scale one. The encoder model p_c is given by Equation (5.4), with the restriction that identical feature functions $\varphi_{jm}(\boldsymbol{\lambda}_f) = \varphi_j(\boldsymbol{\lambda}_f)$ are employed in every coarse scale element and shared coefficients $\tilde{\theta}_{c,jm} = \tilde{\theta}_{c,j}$ are used. As a prior, the RVM-based model presented in Section 5.3.3 is chosen.

The decoder model p_{cf} is given by the shape function interpolation of the coarse grained model, $u_{f,i} = W_{ij} u_{c,j} + \tau_{cf,i}^{-1/2} Z_i$, where the interpolation matrix for linear elements in $1D$ is given by

$$W_{ij} = \begin{cases} \frac{x_i - X_{j-1}}{X_j - X_{j-1}} & \text{for} \quad X_{j-1} \leq x_i \leq X_j, \\ \frac{x_i - X_{j+1}}{X_j - X_{j+1}} & \text{for} \quad X_j \leq x_i \leq X_{j+1}, \\ 0 & \text{else,} \end{cases} \tag{6.2}$$

where x_i, X_i are the coordinates of the vertices of the fine and coarse model, respectively. The precision parameters $\tau_{cf,i}$ are learned from the data and updated in closed form by finding the roots of the gradient specified in Equation (5.30).

Validation purpose

In $1D$ Darcy flow, it is known [1] that the effective permeability is given in closed form by the harmonic mean defined by Equation (5.15). Among 99 other feature functions of the various types discussed in Section 5.2.1, the harmonic mean over the 8 fine scale elements of each coarse element is therefore included as a feature $\varphi(\boldsymbol{\lambda}_f)$ and it is expected that it is the only one being activated by the sparsity prior model.

Results

The described model is trained with $N = 16$ training samples and the true, fine scale solution $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$ for a test sample $\boldsymbol{\lambda}_f$ that is distributed as described above is

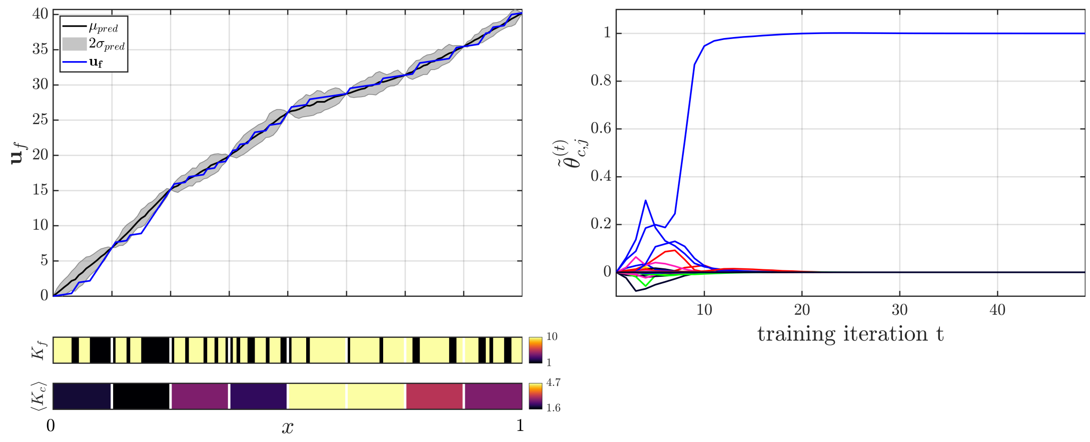


FIGURE 6.1: One-dimensional example. Left: True FGM solution $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$ (blue line) for the $1D$ microstructure $\boldsymbol{\lambda}_f$ depicted in the top bar shown below. The black line + gray shaded areas show the surrogate mean prediction $\boldsymbol{\mu}_{\text{pred}} \pm$ two standard deviations $\boldsymbol{\sigma}_{\text{pred}}$. The lowermost bar shows the mean effective permeability $\langle \boldsymbol{\lambda}_c \rangle$ field. Right: Convergence of feature function coefficients $\tilde{\boldsymbol{\theta}}_c$ over training iterations t. It can nicely be observed that all features except for one are pruned by the sparsity RVM based prior. The only nonzero $\tilde{\theta}_{c,j}$ corresponds to the harmonic mean and converges to 1, as expected. Picture taken from [104].

depicted in the left picture of Figure 6.1 (blue line) together with the predictive distribution (black line and grey shaded area) of the surrogate model. The microstructure $K_f(x, \boldsymbol{\lambda}_f)$ and the expected effective permeability $\langle K_c(x, \boldsymbol{\lambda}_c) \rangle$ are shown in the colored bars below.

The right side of Figure 6.1 shows the convergence of the feature function coefficients $\tilde{\boldsymbol{\theta}}_c$ over training iterations t. As expected, all feature coefficients $\tilde{\theta}_{c,j}$ go to 0 except for the one corresponding to the harmonic mean feature function, which converges to 1.

The predictive performance for the model trained on $N = 16$ FGM runs is assessed on a test set comprising $N_{\text{test}} = 1024$ samples. We obtain

$$e = 0.027(3) \tag{6.3}$$

being approximately 30 times smaller than the reference value 1. It can be observed that even with a perfect feature function such as the harmonic mean in $1D$, there will always be predictive errors/uncertainties due to the information loss taking place during the coarse graining process.

6.1.2 Two-dimensional examples

All of the following experiments using Darcy flow FGM data are conducted in two spatial dimensions where no closed form solution for the effective permeability field encoded by $\boldsymbol{\lambda}_c$ is available. The FGM is given by a standard Galerkin finite element

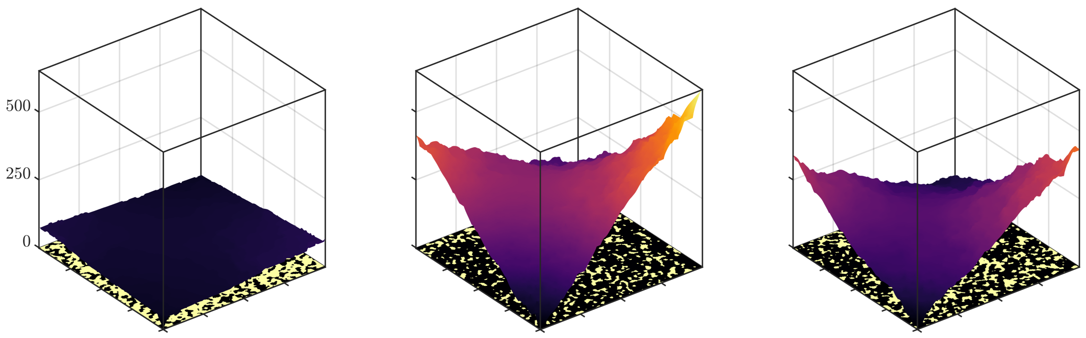


FIGURE 6.2: Two-dimensional Darcy flow examples. Microstructures are generated as described in Section 6.1.2 with $c = \frac{K_{hi}}{K_{lo}} = 100$ and boundary conditions are set according to $\boldsymbol{a} = (0,\ 800,\ 1200,\ -2000)^T$. Picture taken from [104].

model for the PDE defined in Equation (3.3) discretized on a 256×256 square mesh of bilinear finite elements. Each of the square finite elements (or pixels) is assigned a constant, isotropic binary permeability K_{hi} or K_{lo}. The microstructural description $\boldsymbol{\lambda}_f$ therefore consists of a list of all pixel values and the input dimension is $\dim(\boldsymbol{\lambda}_f) = 256 \times 256 = 65536$. The FGM output $\boldsymbol{u}_f$ is given by the vertex values[1] of the pressure field, $u_{f,i} = P(\boldsymbol{x}_i)$, hence the output dimension is $\dim(\boldsymbol{u}_f) = 257 \times 257 = 66049$.

Again, the microstructures are generated by sampling from a zero-mean Gaussian process

$$g(\boldsymbol{x}) \sim GP(0, k(\boldsymbol{x} - \boldsymbol{x}')) \tag{6.4}$$

with stationary covariance function $k(\boldsymbol{x} - \boldsymbol{x}') = \exp\left\{-\frac{(\boldsymbol{x}-\boldsymbol{x}')^2}{l^2}\right\}$, $l = 0.01$ in conjunction with a cutoff parameter c_{cut} controlling the volume fraction which is randomized as $|\Omega_f| \sim U(0,1)$ as discussed in Section 2.4. The weakly permeable phase is fixed to have permeability $K_{lo} = 1$ so that highly permeable phase permeability $K_{hi} = 2 \ldots 1000$ controls the contrast ratio $c = \frac{K_{hi}}{K_{lo}}$. Boundary conditions are set according to

$$P_{bc}(\boldsymbol{x}) = a_0 + a_x x_1 + a_y x_2 + a_{xy} x_1 x_2, \qquad \boldsymbol{x} \in \Gamma_P, \tag{6.5}$$

$$\boldsymbol{V}_{bc}(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} P_{bc}(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma_V \tag{6.6}$$

since boundary conditions of this functional form imply minimal discretization error given the bilinear FEM shape functions. Indicative FGM samples with contrast $c = \frac{K_{hi}}{K_{lo}} = 100$ and boundary conditions corresponding to $\boldsymbol{a} = (0,\ 800,\ 1200,\ -2000)^T$ and $\Gamma_P = \{\boldsymbol{0}\}$, $\Gamma_V = \partial\Omega \backslash \Gamma_P$ can be seen in Figure 6.2. A single FGM evaluation $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$ (i.e., matrix assembly and system solution) takes us 23(4)s on an Intel Xeon E5-2650 2.00GHz CPU.

[1]The vertex values are identical to the degrees of freedom in this finite element model.

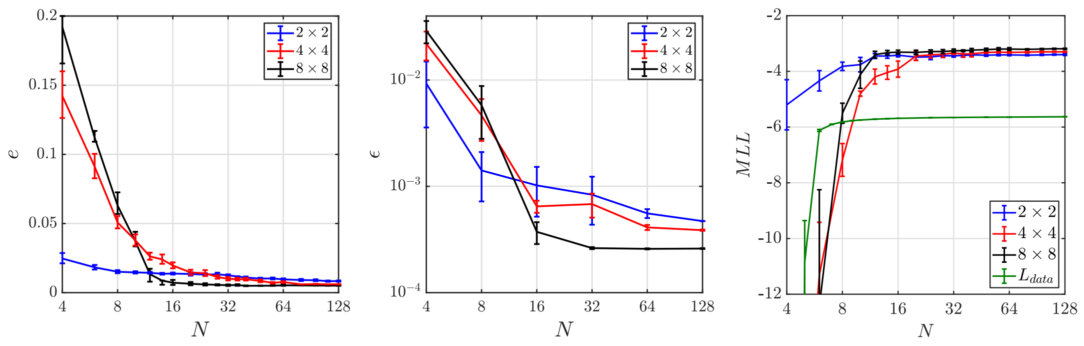


FIGURE 6.3: Predictive performance metrics e, ϵ and MLL as defined in Equations Section 5.4.1 in dependence of training data N and CGM resolution $\dim(\boldsymbol{\lambda}_c)$. A test set with $N_{\text{test}} = 1024$ samples drawn from the same distribution as the training set is used, such that the indicated error bars are solely due to randomization of training sets. Picture adapted from [104].

6.1.3 Modeling considerations

In the following experiments, a Darcy flow CGM with a regular square grid permeability field discretization of resolution 2×2, 4×4, and 8×8 is employed. The PDE discretization is identical to the permeability field discretization and bilinear shape functions ψ_j are used. A single CGM evaluation $\boldsymbol{u}_c(\boldsymbol{\lambda}_c)$ (matrix assembly and system solution) takes 0.13(6)ms, 0.22(5)ms, and 0.99(1)ms of computation time, depending on the PDE discretization resolution. The coarse-to-fine projection matrix $\boldsymbol{W} \in \mathbb{R}^{\dim(\boldsymbol{u}_f) \times \dim(\boldsymbol{u}_c)}$ of the decoder distribution p_{cf} is set to the coarse model shape function interpolant, i.e., $W_{ij} = \psi_j(\boldsymbol{x}_i)$, where $\boldsymbol{x}_i$ are CGM vertex coordinates. All other model parameters, i.e., $\boldsymbol{\tau}_{cf}$, $\boldsymbol{\tau}_c$, and $\tilde{\boldsymbol{\theta}}_c$ are trained according to the RVM-based prior model as described in Section 5.3.3 and the restriction $\tilde{\theta}_{c,jm} = \tilde{\theta}_{c,j}$ is employed. The feature functions employed in this model are summarized in Appendix A.

6.1.4 Predictive performance

We compute the performance metrics e, ϵ and MLL as discussed in Section 5.4.1 for models trained on $N = 4$ up to $N = 128$ FGM evaluations on a test set of $N_{\text{test}} = 1024$. Monte Carlo errors on the performance error measures e, ϵ and MLL can therefore be neglected. The FGM data (both for training and testing) is generated as described in Section 6.1.2, with a contrast ratio $c = K_{hi}/K_{lo} = 2$ and boundary conditions defined by $\boldsymbol{a} = (0,\ 800,\ 1200,\ -2000)^T$ and $\Gamma_P = \{\boldsymbol{0}\}, \Gamma_V = \partial\Omega \backslash \Gamma_P$.

The dependence of the model predictive quality on the training data N and on the CGM resolution is depicted in Figure 6.3. It is observed that all three performance measures reach their asymptotic values with $N \gtrsim 32$ training samples, irrespective of the CGM discretization resolution. Coarser CGM models tend to be more stable for few training data $N \lesssim 10$, but in the high data limit, finer discretized CGMs lead to better performance. It is noted that the performance metrics also depend on the particular training samples chosen. The model is therefore trained multiple times

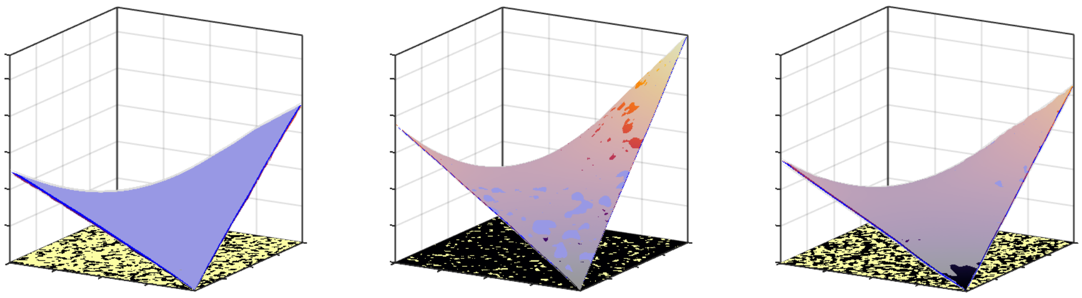


FIGURE 6.4: Predictive mean μ_{pred} (blue) $\pm\sigma_{\text{pred}}$ (transparent gray) together with the true FGM response $u_f(\lambda_f)$ (colored) and the corresponding permeability field $K(x, \lambda_f)$ (bottom surface) with $c = \frac{K_{hi}}{K_{lo}} = 2$ and a CGM discretization of 8×8. The model is trained on $N = 128$ FGM evaluations. Picture taken from [104].

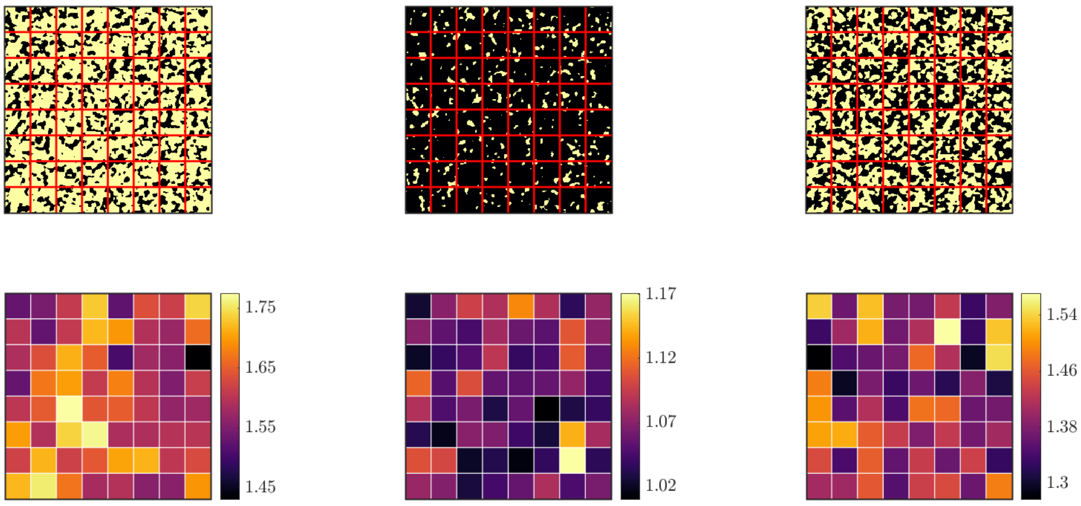
with varying training sets and averages of e, ϵ and MLL together with their standard Monte Carlo error are computed, which is indicated by the error bars in Figure 6.3. Figure 6.4 shows three indicative predictions on test samples for $c = K_{hi}/K_{lo} = 2$, $N = 128$ and a 8×8 CGM resolution (both PDE and permeability field). The true solution (colored) is tightly enveloped by the predictive distribution visualized by the predictive mean μ_{pred} (blue surface) $\pm\sigma_{\text{pred}}$ (transparent gray).

6.1.5 Effective CGM permeability field

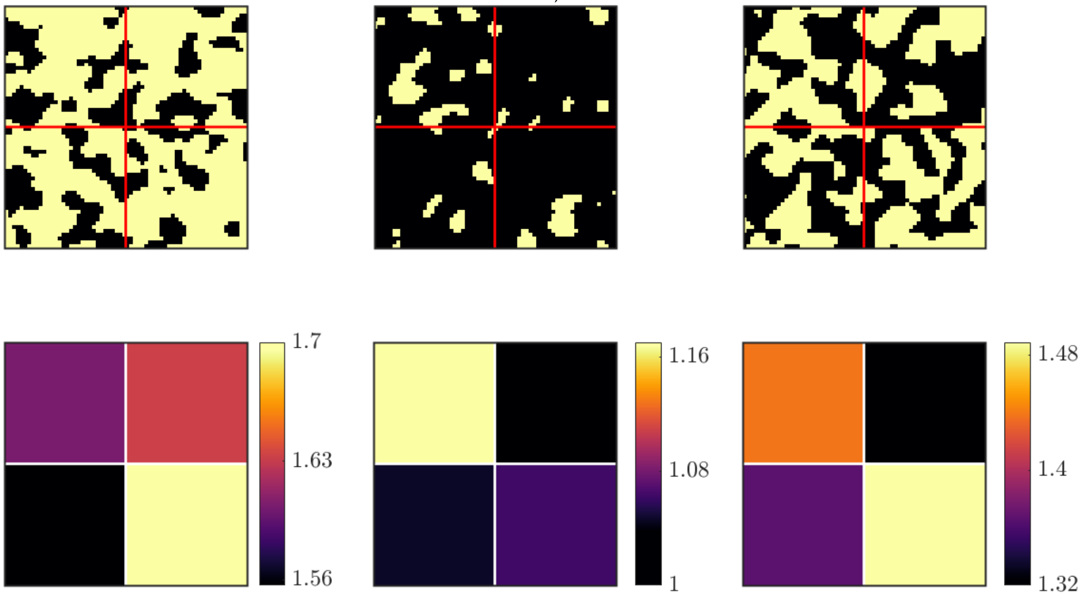
Further physical insight is given by 6.5 where the effective permeability field K_{eff} of the three test samples of 6.4 is depicted alongside with the true FGM permeability field $K(x, \lambda_f)$. It can be observed that subregions Ω_m with predominantly weakly permeable pixels have low effective permeability and vice versa. Moreover, it is observed that $K_{lo} \leq K_{\text{eff}} \leq K_{hi}$, as one would expect.

6.1.6 Activated feature functions

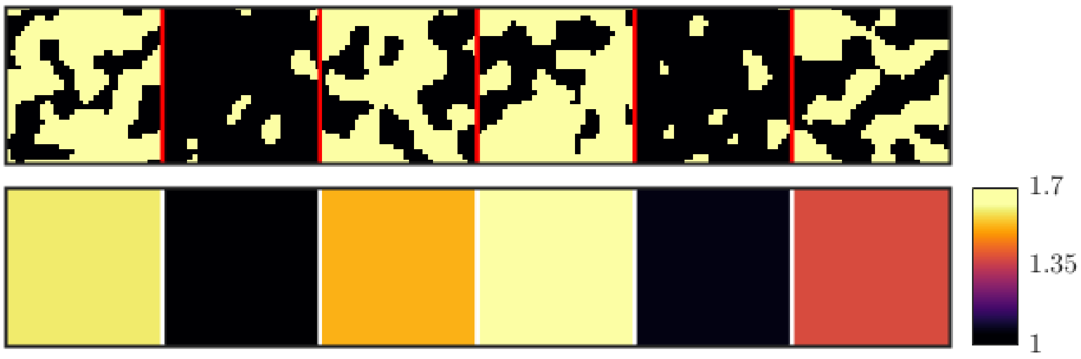
From a physical point of view, it may be interesting to see which feature functions $\varphi_j(\lambda_f)$ are activated for different contrast values $c = K_{hi}/K_{lo}$. To address this question, a model with the identical 100 feature functions as before and a 4×4 CGM solver is trained with data of contrast values $c = 2, 10, 100, 500, 1000$. As the primary objective in this experiment is not optimal predictive performance under minimal training data but interpretation of feature functions, a large training set of $N = 1024$ is used to suppress effects of finite training data. Furthermore, to suppress effects of boundary conditions on the activated feature functions, the coefficients a of Equation (6.5) are randomized like $a \sim \mathcal{N}(\mathbf{0}, \text{diag}(\sigma_a^2))$ with $\sigma_a^2 = (0,\ 10^6,\ 10^6,\ 10^6)^T$. The feature function coefficient MAP estimates $\tilde{\theta}_{c,jm} = \tilde{\theta}_{c,j}$ for the different contrast values are shown in Figure 6.6 and clearly exhibit sparsity. It is observed that for higher contrast values, the number and magnitude of activated coefficients $\tilde{\theta}_{c,j}$ increases. Moreover, feature functions evaluated over the whole domain Ω instead of single subregions Ω_m become more prominent for increasing contrast values. Most



(A) Coarse-grained, effective permeability $\langle K_{\text{eff},m} \rangle$ for the three test samples shown in Figure 6.4 with $N = 128$, $N_{el,c} = 8 \times 8$.



(B) Lower right corner macro-cells and mean effective permeabilities $\langle K_{\text{eff},m} \rangle$ of the microstructures shown in 6.5a.



(C) Effective permeabilities of randomly chosen macro-cells of the microstructures shown in 6.5a.

FIGURE 6.5: Illustrations of the predictive posterior mean permeability $\langle K_{\text{eff},m} \rangle = \langle e^{\lambda_{c,m}} \rangle$. Pictures taken from [104].

relevant features are generalized means, effective medium approximations (differential effective medium, self-consistent approximation), the "Gaussian linear filter"

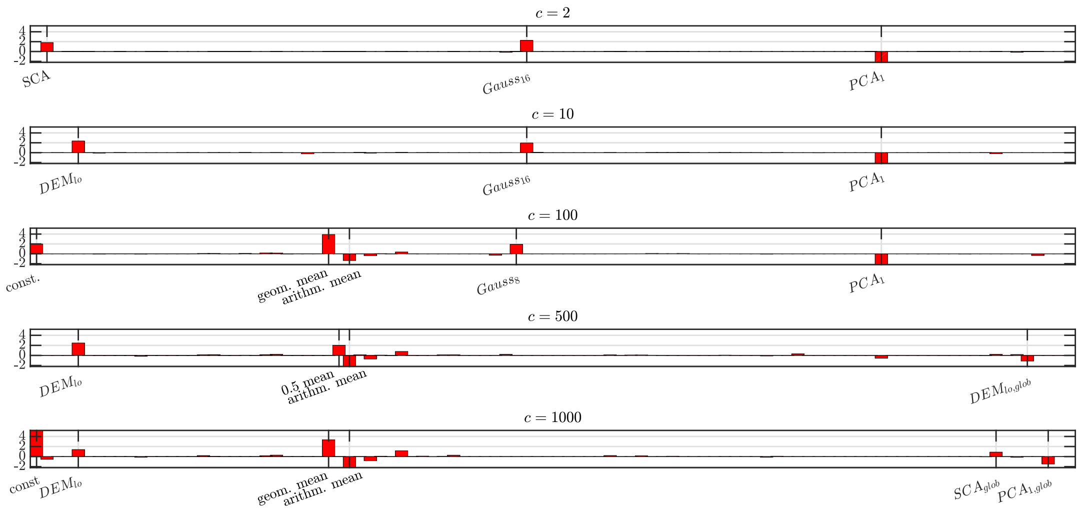


FIGURE 6.6: MAP estimates of the feature function coefficients $\tilde{\boldsymbol{\theta}}_c$ for different contrast values. Picture taken from [104].

and the first principal component (found on using 4096 samples of $\boldsymbol{\lambda}_f$). For feature function explanation, see Section 5.2.1.

6.1.7 Extrapolative capabilities: variation of boundary conditions

A convenient property of the proposed surrogate modeling framework is that it is possible to deterministically imprint the boundary conditions of the FGM data on the CGM model. Therefore, having trained the surrogate on FGM training data which have, say, boundary conditions defined by the coefficient vector $\boldsymbol{a}$, it is possible to carry out predictions under a modified set of boundary conditions, say $\tilde{\boldsymbol{a}} \neq \boldsymbol{a}$, without significant deterioration of predictive quality. This implies that the surrogate is capable to *extrapolate* well on data which are very different from the data seen during training, showing that the surrogate correctly encodes the salient physical a priori information given by boundary conditions. Such a property is typically not implied by off-the-shelf machine learning algorithms which exclusively model the input-output relation of the data [46, 72].

To show this feature of the proposed surrogate, a model based on a 4×4 CGM solver is trained on $N = 1024$ FGM evaluations (to avoid effects of small data), once using boundary conditions defined by $\boldsymbol{a} = (0,\ 800,\ 1200,\ -2000)^T$, and once using $\tilde{\boldsymbol{a}} = (0,\ 500,\ -1500,\ ,1000)^T$. After training, we use both models for prediction on two $N_{\text{test}} = 1024$ test sets (s.t. Monte Carlo error is negligible), one with boundary conditions $\boldsymbol{a}$, the other one with $\tilde{\boldsymbol{a}}$. Not only do we perform direct predictions, i.e., "train on $\boldsymbol{a}$, ($\tilde{\boldsymbol{a}}$), predict on $\boldsymbol{a}$, ($\tilde{\boldsymbol{a}}$)", but also cross-over predictions like "train on $\boldsymbol{a}$, ($\tilde{\boldsymbol{a}}$), predict on $\tilde{\boldsymbol{a}}$, ($\boldsymbol{a}$)". The resulting measurements for the error metrics e and MLL are displayed in Table 6.1. Only slight deterioration is observed when predictions are carried out on boundary conditions different from the ones used in the training set. One indicative prediction example for every boundary condition combination

Error measure $\langle e \rangle$

trained on / prediction on	$\boldsymbol{a} = \begin{pmatrix} 0 & 800 & 1200 & -2000 \end{pmatrix}^T$	$\tilde{\boldsymbol{a}} = \begin{pmatrix} 0 & 500 & -1500 & 1000 \end{pmatrix}^T$
$\boldsymbol{a} = \begin{pmatrix} 0 & 800 & 1200 & -2000 \end{pmatrix}^T$	0.00895	0.00963
$\tilde{\boldsymbol{a}} = \begin{pmatrix} 0 & 500 & -1500 & 1000 \end{pmatrix}^T$	0.0102	0.00950

Error measure $\langle MLL \rangle$

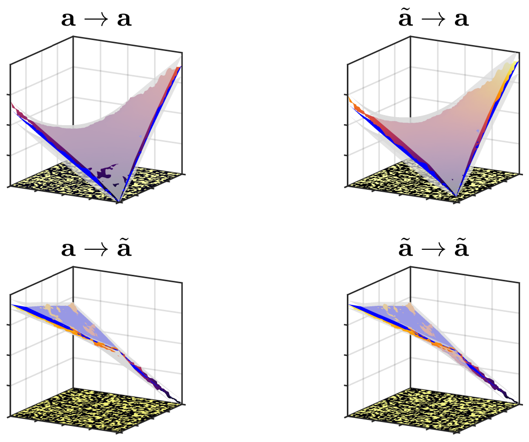
trained on / prediction on	$\boldsymbol{a} = \begin{pmatrix} 0 & 800 & 1200 & -2000 \end{pmatrix}^T$	$\tilde{\boldsymbol{a}} = \begin{pmatrix} 0 & 500 & -1500 & 1000 \end{pmatrix}^T$
$\boldsymbol{a} = \begin{pmatrix} 0 & 800 & 1200 & -2000 \end{pmatrix}^T$	-4.06	-4.31
$\tilde{\boldsymbol{a}} = \begin{pmatrix} 0 & 500 & -1500 & 1000 \end{pmatrix}^T$	-3.90	-3.86

TABLE 6.1: Averaged predictive quality measures e and MLL as defined in Equation (5.85) and Equation (5.87). Predictions are only slightly worse when the model is used for predictions on boundary conditions that are different from the ones used in the training data. Table adapted from [104].

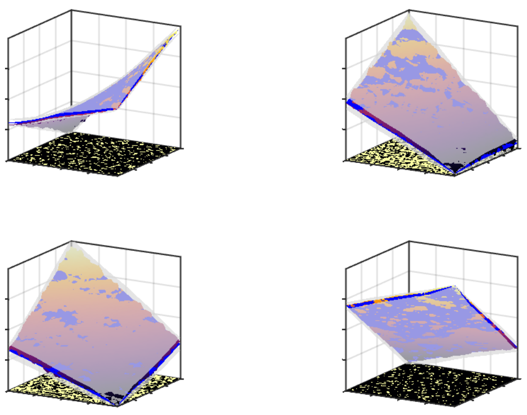
is shown in Figure 6.7a. Figure 6.7b shows predictions on boundary conditions randomized as $\tilde{\boldsymbol{a}} \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\sigma^2_{\tilde{a}}))$, $\sigma^2_{\tilde{a}} = (0,\ 10^6,\ 10^6,\ 10^6)^T$ where the surrogate is trained using boundary conditions as defined by $\boldsymbol{a}$ above. The true solutions of the test samples are always well included in the predictive distributions.

6.2 Different physics: Stokes flow FGM data

This section contains numerical experiments where the FGM training and test data are generated from a Stokes flow finite element simulation as described in Section 3.1.2. As outlined in Section 2.3, in the scale separation limit $\ell_f \ll r_0 \ll L$, where ℓ_f is the characteristic length scale of the microstructural pore space, r_0 the radius of volume averaging and L the length scale of the domain, Stokes flow (Equation (2.5)) can accurately be described by Darcy's equations of motion (Equation (2.6a)). It is therefore possible to apply the surrogate model proposed in Chapter 5 to Stokes flow problems as well, where the CGM unit remains a simplified Darcy flow solver operating on very coarse discretizations. It is clear that full Stokes to Darcy transition may only be observed in the infinite scale separation limit. Nevertheless, Darcy's equations can still be used as a stencil of a machine learning model for Stokes flow FGM data very far from scale separation due to the model's parametric adaptivity. Moreover, under the worst-case assumption that a coarse-grained Darcy flow simulation would lose all information of the underlying Stokes flow pressure and velocity field, the surrogate is safeguarded to misuse because overly large predictive uncertainty σ_{pred} would directly signal low predictive quality. The experimental results presented here are based on the VRVM prior model discussed in Section 5.3.4 and have been published in [87].



(A) Predictive mean μ_{pred} (blue) $\pm\sigma_{\text{pred}}$ (transparent grey) and true response $\boldsymbol{u}_f$ (colored). The plots show indicative examples for predictions with boundary conditions as specified on top of each subfigure



(B) Predictive mean μ_{pred} (blue) $\pm\sigma_{\text{pred}}$ (transparent grey) and true response $\boldsymbol{u}_f$ (colored). The model is trained using boundary conditions fixed to $\boldsymbol{a}$. Predictions are carried out on a test set with randomized boundary conditions as described in the text.

FIGURE 6.7: Prediction examples for $N = 1024$, $N_{el,c} = 4 \times 4$, $l = 0.01$ and $c = 10$ under variation of boundary conditions in the test data. It is observed that the predictive quality does hardly deteriorate when predicting on data with other boundary conditions as used in the training set. Picture taken from [104].

6.2.1 Fine scale data and computational considerations

As presented in Section 2.1, the porous microstructure of a random medium is described mathematically by a perforated domain, i.e., the domain where Stokes flow is solved is the *fluid* or *pore space* Ω_f, whereas the *solid space* denoted by $\Omega_s = \Omega \backslash \Omega_f$ does **not** correspond to the mathematical domain of the Stokes flow PDE. On the interface of solid and fluid spaces, the no-slip boundary condition $\boldsymbol{V} = 0$ is applied. To obtain a unique solution for the pressure response field $P(\boldsymbol{x})$, it is necessary that the pore space Ω_f is fully connected, which we generate here using non-overlapping polydisperse spherical exclusions as described in Section 2.4.2, see Figure 2.5 for some examples.

The random center coordinates and radii of every exclusion of a microstructural sample n are then passed to the FEniCS mshr module [137] and a triangular finite element mesh of approximately $256 \times 256 = 65,536$ vertices is constructed by constructive solid geometry. Depending on the specific topology of the microstructure, in particular the number of exclusions $N_{\text{ex}}^{(n)}$, the finite element mesh generation requires a significant amount of computation time. On a single Intel Xeon E5-2620 (2.00 GHz) CPU, we require $\approx$ 2 hours of computation for a mesh with $N_{\text{ex}}^{(n)} \approx 1,000$ exclusions, whereas up to 10 days are needed for microstructures with $N_{\text{ex}}^{(n)} \approx 20,000$. The solution of the PDE itself is comparably cheap and requires $1512s \pm 5.7s$ on average on the same hardware.

Similar to the previous examples where FGM and CGM solvers are based on the same governing equations, we use boundary conditions of the form

$$P_{bc}(\boldsymbol{x}) = 0, \qquad \text{for} \qquad \boldsymbol{x} \in \Gamma_P = \boldsymbol{0}, \tag{6.7}$$

$$\boldsymbol{V}_{bc}(\boldsymbol{x}) = \begin{pmatrix} a_x + a_{xy}x_2 \\ a_y + a_{xy}x_1 \end{pmatrix}, \qquad \text{for} \qquad \boldsymbol{x} \in \Gamma_V = \partial\Omega \backslash \boldsymbol{0}, \tag{6.8}$$

compare Equation (6.5). The coefficients $\boldsymbol{a}$ may again assume different values and are therefore specified later.

6.2.2 Modeling considerations

As already stated, the CGM is a finite element solver based on Darcy's equations of motion discretized on a 16×16 square mesh with bilinear shape functions $\psi_j(\boldsymbol{x})$ on the unit square domain $\Omega = [0,1]^2$. The number of degrees of freedom of the CGM is therefore $\dim(\boldsymbol{u}_c) = 17 \cdot 17 - 1 = 288$ (-1 because of the essential boundary condition at $\boldsymbol{x} = \boldsymbol{0}$) and a single CGM evaluation takes $(1.04 \pm 2) \times 10^{-3}s$ on a single Intel Xeon E5-2620 (2.00 GHz) CPU. For a surrogate predictive distribution $N_{\text{samples}} \approx 100$ CGM evaluations are needed, which is to be compared to the FGM solution + mesh generation time stated above. In all of the following examples, the PDE discretization is left unchanged, whereas the effective permeability field may exhibit different discretizations specified later. Again, the CGM permeability field $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c)$ is assumed to have the form defined by Equations (5.7), (5.8). We fix the coarse-to-fine projection matrix $\boldsymbol{W}$ to be the CGM shape function interpolation, $W_{ij} = \psi_j(\boldsymbol{x}_i)$, where $\boldsymbol{x}_i$ are CGM vertex coordinates. The remaining model parameters $\boldsymbol{\tau}_{cf}, \boldsymbol{\tau}_c, \tilde{\boldsymbol{\theta}}_c$ are considered to be latent variables endowed with priors in the framework of the VRVM model presented in Section 5.3.4, under the restriction that the precisions $\tau_{p,jm}$ pertaining to the feature function coefficients $\tilde{\theta}_{c,jm}$ are restricted as $\tau_{p,jm} = \tau_{p,j}$. The set of feature functions employed in the following experiments is summarized in Table B.1 of Appendix B.

6.2.3 Validation: Scale separation/homogenization limit

It was shown in [114] that in the limit of infinite scale separation $\ell_f \ll r_0 \ll L$, Stokes' and Darcy's equations of motion are equivalent. Moreover, the microstructural permeability can be assumed to be homogeneous over the averaging volume of radius r_0. It is therefore expected that, given microstructures of homogeneous permeability on the scale of discretization of the CGM, the predictions of the surrogate model are extraordinarily good because the CGM itself is a very accurate description of the flow problem. The example of this subsection therefore aims to generate FGM samples that are in the scale separation/homogenization limit as much as computationally affordable, whilst showing large variability, e.g., by high values of $\sigma_{\text{ex}}^2, \sigma_r^2$. We therefore set the parameters of the distribution of microstructures $p(\boldsymbol{\lambda}_f)$ to (see

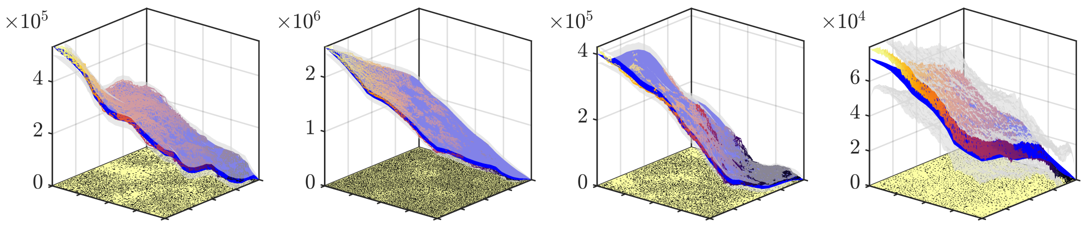


FIGURE 6.8: Predictive examples on FGM data in the homogenization/scale separation limit as described in Section 6.2.3. The colored surfaces represent the true FGM pressure field solution $P(\boldsymbol{x})$, whereas the blue one represents the predictive mean $\boldsymbol{\mu}_{\text{pred}}$ and the transparent gray surfaces depict $\pm\sigma_{\text{pred}}$. Note the high variation on the z-axis. One can also observe that the higher the number of exclusions $N_{\text{ex}}^{(n)}$ is, the better the predictions are. This is because the pore space length scale ℓ_f is smaller for higher $N_{\text{ex}}^{(n)}$, so that we are further in the scale separation limit. Picture taken from [87].

Section 2.4.2)

$$\begin{aligned} \mu_{\text{ex}} &= 8.35, & \sigma_{\text{ex}} &= 0.6, \\ \mu_r &= -5.53, & \sigma_r &= 0.5, \\ \ell_s &= 1.5, & \ell_x &= 0.1, \end{aligned} \tag{6.9}$$

and the boundary conditions are fixed to

$$a_x = a_y = 1, \quad a_{xy} = 0. \tag{6.10}$$

The surrogate model performance is evaluated using three separate training sets of $N = 128$ each. After training on a surrogate using a CGM with regular 4×4 permeability field discretization, the performance metrics R^2 and MLL as introduced in Section 5.4.1 are evaluated on a test set of size $N_{\text{test}} = 1024$ and averaged over the different training sets. We measure

$$R^2 = 0.995 \pm 0.004, \qquad MLL = -11.003 \pm 0.013, \tag{6.11}$$

which means that the surrogate model predictions are very accurate, as expected in the scale separation/homogenization limit. Predictions on four randomly chosen test samples are depicted in Figure 6.8.

It has to be noted that in the limit of infinite scale separation and a homogeneous permeability field over the whole unit square domain $\Omega = [0,1]^2$, the pressure response $P(\boldsymbol{x})$ depicted in Figure 6.8 would correspond to a perfect plane defined by the boundary conditions $a_x = a_y = 1$, $a_{xy} = 0$ and the constant effective permeability $K(\boldsymbol{x}) = K$. In this extremal case, it becomes clear that the effective output dimension is very low and a reduced-basis approach as presented in Section 4.5.4 may be more appropriate. The variance explained by the first few PCA components

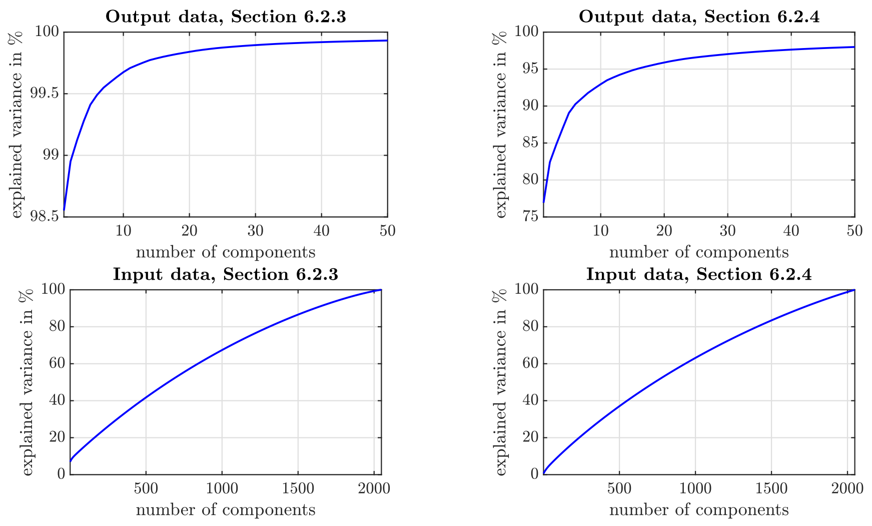


FIGURE 6.9: Variance explained by the most significant PCA components of output data $\boldsymbol{u}_f$ (top row) and input data $\boldsymbol{\lambda}_f$ (bottom) for the FGM data of the examples presented in Sections 6.2.3 (left) and 6.2.4 (right). All plots are obtained by interpolating/discretizing on a regular $257 \times 257/256 \times 256$ square grid (for $\boldsymbol{u}_f/\ \boldsymbol{\lambda}_f$) and then performing PCA on 2048 FGM samples $\boldsymbol{\lambda}_f^{(n)}, \boldsymbol{u}_f^{(n)}$ with microstructure distribution $p(\boldsymbol{\lambda}_f)$ and boundary conditions $\boldsymbol{a}$ as described in the text. Picture adapted from [87].

of the input and output data is plotted in the left half of Figure 6.9. It can be seen that almost all variance of the output data is concentrated in a single PCA component.

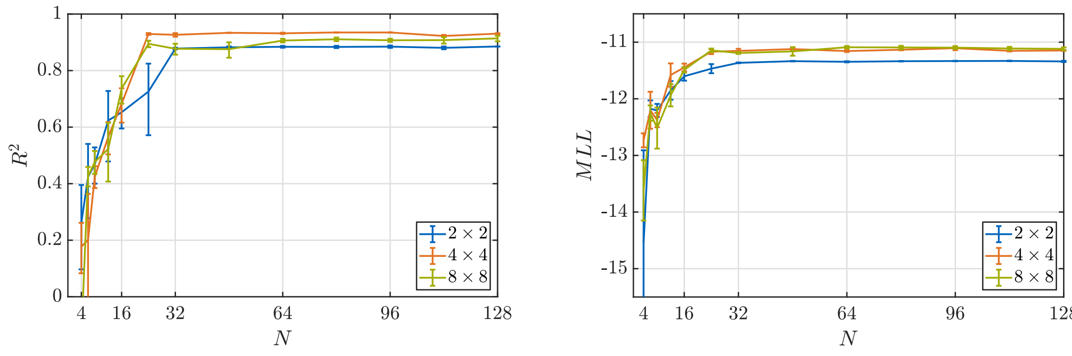


FIGURE 6.10: Performance metrics coefficient of determination R^2 and mean log likelihood MLL as explained in Section 5.4.1 in dependence of the number of training data N and the CGM permeability field resolution $\dim(\boldsymbol{\lambda}_c)$. Performance metrics are averaged over a test set of $N_{\text{test}} = 1024$ samples. The error bars show the Monte Carlo error due to randomization of training data. Picture taken from [87].

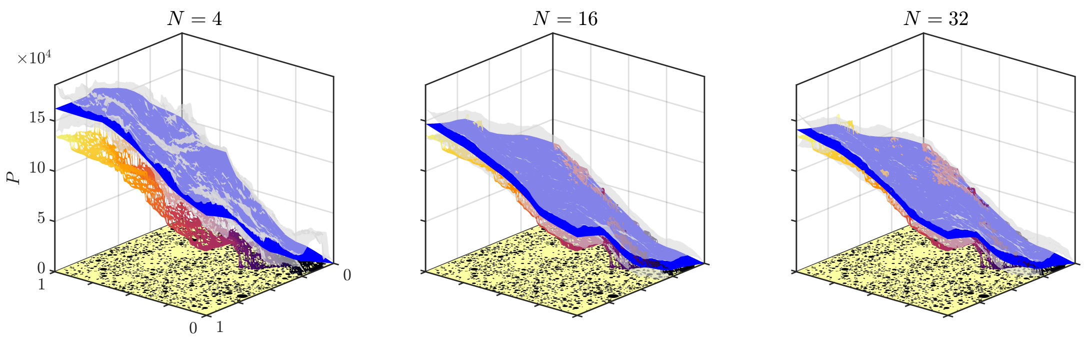


FIGURE 6.11: Predictive examples on the same FGM sample $\boldsymbol{\lambda}_f, \boldsymbol{u}_f$ sampled as described in Section 6.2.4 using a 4×4 effective permeability field discretization CGM and different number of training data N. The colored surface is the true solution of the test sample, the blue one shows the predictive mean $\boldsymbol{\mu}_{\text{pred}}$ and the transparent gray surfaces show $\pm\sigma_{\text{pred}}$. As expected, predictive quality improves with increasing number of training data N. Picture taken from [87].

6.2.4 Predictive performance far from scale separation

In this subsection, we evaluate the predictive performance measures R^2 and MLL introduced in Section 5.4.1 for the CGM effective permeability field discretizations 2×2, 4×4, and 8×8 (corresponding to $\dim(\boldsymbol{\lambda}_c) = 4$, $\dim(\boldsymbol{\lambda}_c) = 16$, and $\dim(\boldsymbol{\lambda}_c) = 64$) and different number of training data N drawn according to

$$\begin{aligned} \mu_{\text{ex}} &= 7.8, & \sigma_{\text{ex}} &= 0.2, \\ \mu_r &= -5.23, & \sigma_r &= 0.3, \\ \ell_s &= 1.2, & \ell_x &= 0.08, \end{aligned} \tag{6.12}$$

with identical boundary conditions as in the previous example, i.e., $a_x = a_y = 1$, $a_{xy} = 0$.

Figure 6.10 shows the expected performance metrics R^2 and MLL in dependence

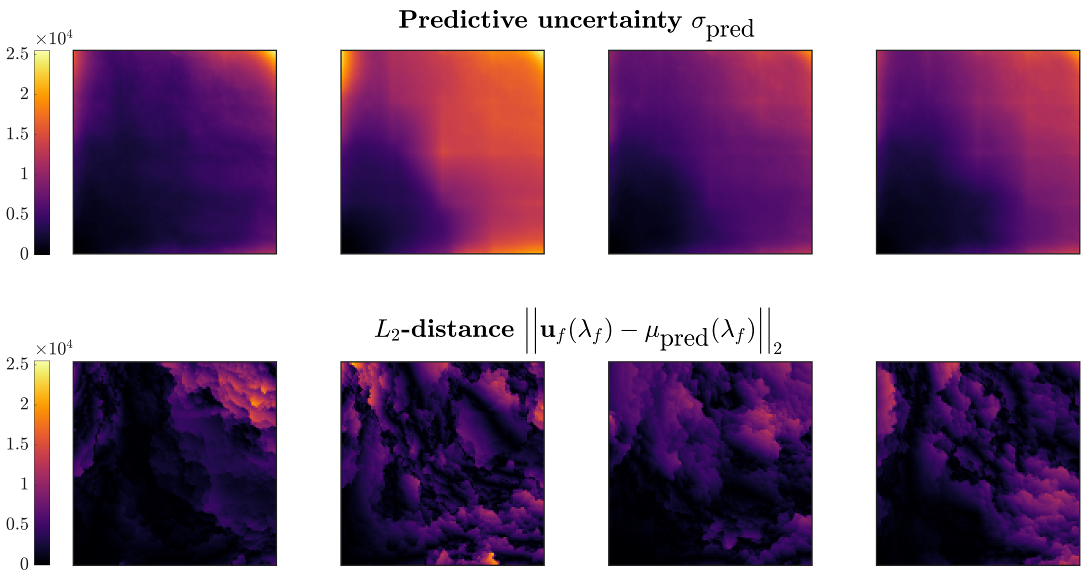


FIGURE 6.12: Predictive uncertainty σ_{pred} (top row) and absolute L_2-error $\|\boldsymbol{u}_f - \boldsymbol{\mu}_{\text{pred}}\|$ of the predictive mean to the true solution for four different test samples. As expected, the predictive error measure σ_{pred} is high in regions where the true L_2 error is high and vice versa. Picture taken from [87].

of the number of training data N and the CGM resolution/latent space dimension $\dim(\boldsymbol{\lambda}_c)$. As in the example of Section 6.1.4, performance metrics are evaluated on a large test set of $N_{\text{test}} = 1024$ FGM evaluations such that errors due to small test sets can be neglected. The error bars shown are Monte Carlo errors due to randomization of training data. It is observed that R^2 and MLL reach their asymptotic values already at $N \approx 32$ training samples. Another interesting observation is that it is not the surrogate with highest resolution CGM (8×8) but the 4×4 model which shows best performance even in the high data limit. This might be due to the higher information loss during the coarse-graining process for finer discretized CGM permeability fields since most feature functions $\varphi_{jm}(\boldsymbol{\lambda}_f)$ are evaluated on the corresponding subregion Ω_m only. Predictive examples on a single microstructure $\boldsymbol{\lambda}_f$ using a 4×4 CGM and different training data N are shown in Figure 6.11.

6.2.5 Predictive uncertainty: σ^2_{pred} and true L_2-error

Additionally to the MLL performance metric, which measures how well the true solution of a test sample is included in the predictive distribution, we have a look at the predictive uncertainty σ_{pred} over the domain Ω and compare it to the 'true' L_2 distance of the predictive point estimate $\boldsymbol{\mu}_{\text{pred}}$ to the true pressure field response represented by $\boldsymbol{u}_f$. To evoke different average velocities in different regions of the domain (and therefore different predictive errors), non-homogeneous flux boundary conditions are imposed by setting $a_x = a_y = 0, a_{xy} = -1$. Both σ_{pred} and the L_2 distance to the true solution are shown for four different samples in Figure 6.12. As expected, it can be observed that σ_{pred} is higher where the true L_2 error is higher.

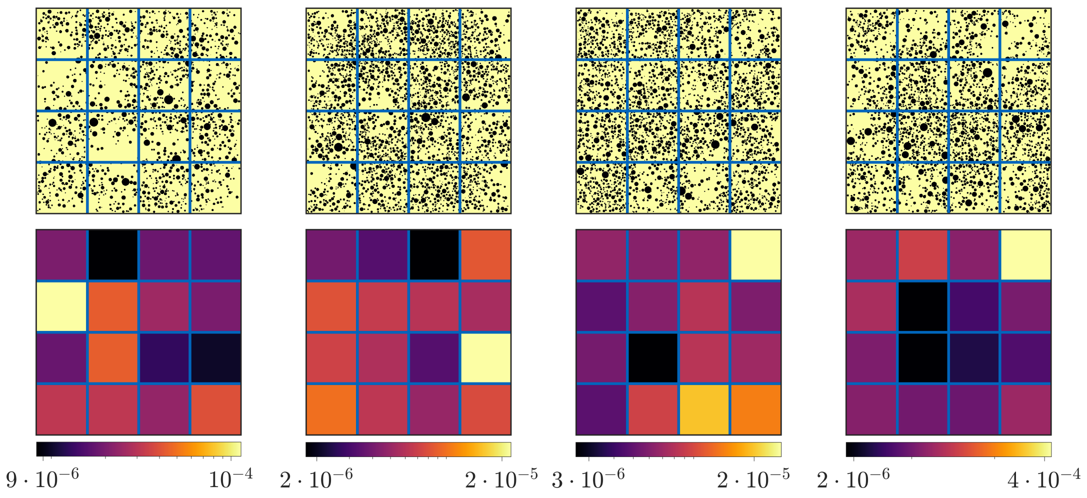


FIGURE 6.13: True porous microstructures encoded by $\boldsymbol{\lambda}_f$ (top row, black = Ω_s, yellow = Ω_f) together with the corresponding mean effective permeability fields $\langle K(\boldsymbol{x}, \boldsymbol{\lambda}_c) \rangle_{Q_{\lambda_c}}$ (bottom) after training with $N = 128$ FGM samples as described in Section 6.2.4. It is clearly visible that cells Ω_m of high pore fraction $|\Omega_{m,f}|/|\Omega_{m,s}|$ have higher effective Darcy flow permeability, as expected. Picture taken from [87].

However, it is impossible to resolve predictive errors on a shorter length scale than the one given by the CGM effective permeability field discretization.

6.2.6 Effective CGM permeability field

The coarse-graining process $\boldsymbol{\lambda}_f \mapsto \boldsymbol{\lambda}_f$ mediated by p_c can be viewed as a process of data-based, numerical homogenization. Further intuition can be obtained by inspecting the mean effective permeability field $\langle K(\boldsymbol{x}, \boldsymbol{\lambda}_c) \rangle_{Q_{\lambda_c}}$ encoded by $\boldsymbol{\lambda}_c$ together with the true microstructures encoded by $\boldsymbol{\lambda}_f$ as shown in Figure 6.13. As expected, the effective permeability is high in cells Ω_m of high pore fraction $|\Omega_{m,f}|/|\Omega_{m,s}|$ and vice versa.

6.2.7 Activated feature functions

The left part ofFigure 6.14 depicts the average number of activated feature functions φ in dependence of the size of the training data N, i.e., the number of nonzero prior feature variances $\gamma_{jm} = \gamma_j$. With increasing training data, more and more feature functions become activated because the importance of the likelihood increases compared to the influence of the prior.

As can be seen in the right part of the figure, for identical FGM data as in Section 6.2.4, the activated features for a training run with $N = 128$ and a 4×4 CGM are a constant bias term, the log self-consistent approximation (infinite contrast limit), the square interface area, and some evaluations of the lineal path function of the pore space Ω_f.

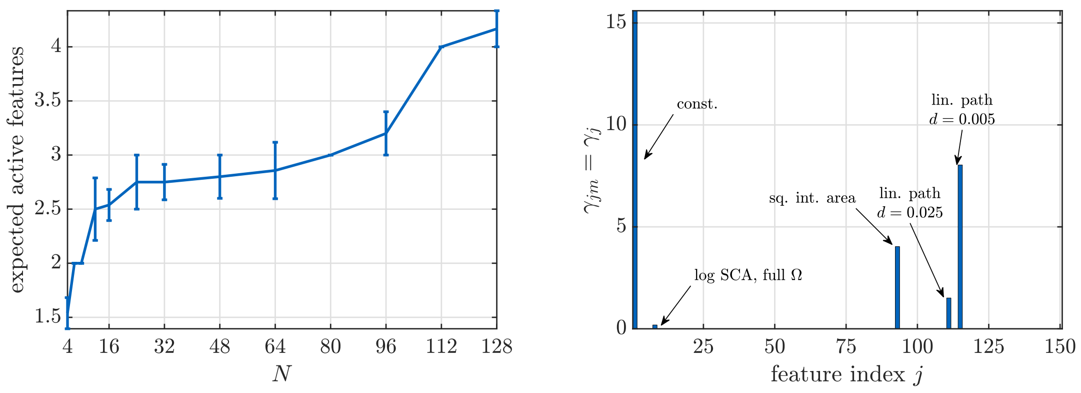


FIGURE 6.14: Expected total number of activated feature functions in dependence of the number of training data N (left). Error bars are due to randomization of training sets. The expected number of activated feature functions is monotonically increasing because the likelihood function becomes more and more important with increasing data compared to the prior. The right side of the figure shows the feature function variances $\gamma_{jm} = \gamma_j = \tau_{p,j}^{-1}$ as an indicator of feature activity. A 4×4 CGM and FGM data as in Section 6.2.4 is used, with $N = 128$ for the example on the right. Picture taken from [87].

6.2.8 Extrapolative capabilities: variation of boundary conditions

As already shown in Section 6.1.7 in the context of Darcy flow FGM data, it is possible to imprint the FGM data boundary conditions to the central CGM unit of the surrogate model. It is thus possible to train the surrogate using FGM data with boundary conditions determined by, say, the coefficient vector $\boldsymbol{a}$ (see Equation (6.8)) but then using it for predictions on FGM data with different pressure and flow boundary conditions determined by $\tilde{\boldsymbol{a}} \neq \boldsymbol{a}$. This suggests that the surrogate is capable to learn the salient microstructural features and use them for extrapolative predictions on data that are very different from the training set.

As an example, boundary conditions of the form $\boldsymbol{a} = \begin{pmatrix} a_x = 1 & a_y = 1 & a_{xy} = 0 \end{pmatrix}^T$ (same as in Section 6.2.4) as well as $\tilde{\boldsymbol{a}} = \begin{pmatrix} \tilde{a}_x = 0 & \tilde{a}_y = 0 & \tilde{a}_{xy} = -1 \end{pmatrix}^T$ are used to train a surrogate with a $\dim(\boldsymbol{\lambda}_c) = 4 \times 4$ CGM permeability field and $N = 128$ FGM samples with identical microstructural distribution as in Section 6.2.4. The measured performance metrics R^2 and MLL for all four training/prediction combinations of boundary conditions $\boldsymbol{a}, \tilde{\boldsymbol{a}}$ evaluated on $N_{\text{test}} = 1024$ test samples and averaged over three different $N = 128$ training sets are summarized in Table 6.2. Only slight deterioration in predictive quality is observed when predictions are carried out on test samples with boundary conditions different from the ones of the training data. Indicative test samples are shown in the left half of Figure 6.15.

One may even go one step further and fully randomize the boundary conditions[2] in the training and test datasets. We generate test and training data with identical

[2]It is noted that, even if the boundary conditions are random, they are fully known to the surrogate model.

Error measure $\langle R^2 \rangle$

trained on / prediction on	$\boldsymbol{a} = \left(a_x{=}1 \; a_y{=}1 \; a_{xy}{=}0 \right)^T$	$\tilde{\boldsymbol{a}} = \left(\tilde{a}_x{=}0 \; \tilde{a}_y{=}0 \; \tilde{a}_{xy}{=}-1 \right)^T$
$\boldsymbol{a} = \left(a_x{=}1 \; a_y{=}1 \; a_{xy}{=}0 \right)^T$	$0.930 \pm 1.3 \cdot 10^{-3}$	$0.863 \pm 1.1 \cdot 10^{-2}$
$\tilde{\boldsymbol{a}} = \left(\tilde{a}_x{=}0 \; \tilde{a}_y{=}0 \; \tilde{a}_{xy}{=}-1 \right)^T$	$0.887 \pm 7.5 \cdot 10^{-3}$	$0.903 \pm 7.0 \cdot 10^{-3}$

Error measure $\langle MLL \rangle$

trained on / prediction on	$\boldsymbol{a} = \left(a_x{=}1 \; a_y{=}1 \; a_{xy}{=}0 \right)^T$	$\tilde{\boldsymbol{a}} = \left(\tilde{a}_x{=}0 \; \tilde{a}_y{=}0 \; \tilde{a}_{xy}{=}-1 \right)^T$
$\boldsymbol{a} = \left(a_x{=}1 \; a_y{=}1 \; a_{xy}{=}0 \right)^T$	$-11.1 \pm 7.6 \cdot 10^{-3}$	$-11.8 \pm 9.3 \cdot 10^{-3}$
$\tilde{\boldsymbol{a}} = \left(\tilde{a}_x{=}0 \; \tilde{a}_y{=}0 \; \tilde{a}_{xy}{=}-1 \right)^T$	$-10.2 \pm 1.3 \cdot 10^{-2}$	$-9.93 \pm 2.3 \cdot 10^{-2}$

TABLE 6.2: Predictive quality metrics R^2 and MLL as defined in Equation (5.86) and Equation (5.87) for predictions on the same and on different boundary conditions as used in the training data. Predictions are only slightly worse when boundary conditions are different from the ones used in the training data. Table adapted from [87].

microstructures as used in Section 6.2.4 and boundary conditions randomized according to $a_x \sim \mathcal{N}(0, 1)$, $a_y \sim \mathcal{N}(0, 1)$, and $a_{xy} \sim \mathcal{N}(0, 1)$. Using $N = 128$ FGM input-output pairs for training, the performance metrics R^2 and MLL are evaluated on a set of $N_{\text{test}} = 1024$ test samples and are found to be

$$R^2 = 0.9776 \pm 0.0027, \qquad MLL = -11.07 \pm 0.089. \tag{6.13}$$

Four random predictive samples are shown in the right part of Figure 6.15.

6.2.9 Uncertainty propagation problem

As a use case of the proposed surrogate modeling framework, a simple uncertainty propagation (UP) problem is considered here. Uncertainty propagation addresses the question of finding/estimating the distribution of a quantity of interest (QoI) $f(\boldsymbol{u}_f)$ given the distribution of model outputs $p(\boldsymbol{u}_f)$,

$$p(\boldsymbol{u}_f) = \int p(\boldsymbol{u}_f | \boldsymbol{\lambda}_f) p(\boldsymbol{\lambda}_f) d\boldsymbol{\lambda}_f, \tag{6.14}$$

where $p(\boldsymbol{u}_f | \boldsymbol{\lambda}_f) = \delta(\boldsymbol{u}_f - \boldsymbol{u}_f(\boldsymbol{\lambda}_f))$ if the expensive FGM solver $\boldsymbol{u}_f(\boldsymbol{\lambda}_f) : \boldsymbol{\lambda}_f \mapsto \boldsymbol{u}_f$ is applied. From the above equation, it follows formally that

$$p(f) = \int \delta(f - f(\boldsymbol{u}_f)) p(\boldsymbol{u}_f) d\boldsymbol{u}_f = \int \delta(f - f(\boldsymbol{u}_f)) p(\boldsymbol{u}_f | \boldsymbol{\lambda}_f) p(\boldsymbol{\lambda}_f) d\boldsymbol{\lambda}_f d\boldsymbol{u}_f \tag{6.15}$$

Needless to say, $p(\boldsymbol{\lambda}_f)$ and the corresponding integration are not given in closed form and evaluation of the FGM $\boldsymbol{u}_f(\boldsymbol{\lambda}_f)$ is costly such that Monte Carlo integration is time-consuming or even impossible. However, sampling from $p(\boldsymbol{\lambda}_f)$ is possible (see Section 2.4) and the distribution $p(\boldsymbol{u}_f | \boldsymbol{\lambda}_f) = \delta(\boldsymbol{u}_f - \boldsymbol{u}_f(\boldsymbol{\lambda}_f))$ can be approximated by the probabilistic surrogate as $p(\boldsymbol{u}_f | \boldsymbol{\lambda}_f) \approx p_{\text{pred}}(\boldsymbol{u}_f | \boldsymbol{\lambda}_f, \mathcal{D})$, where $\mathcal{D}$ is the training

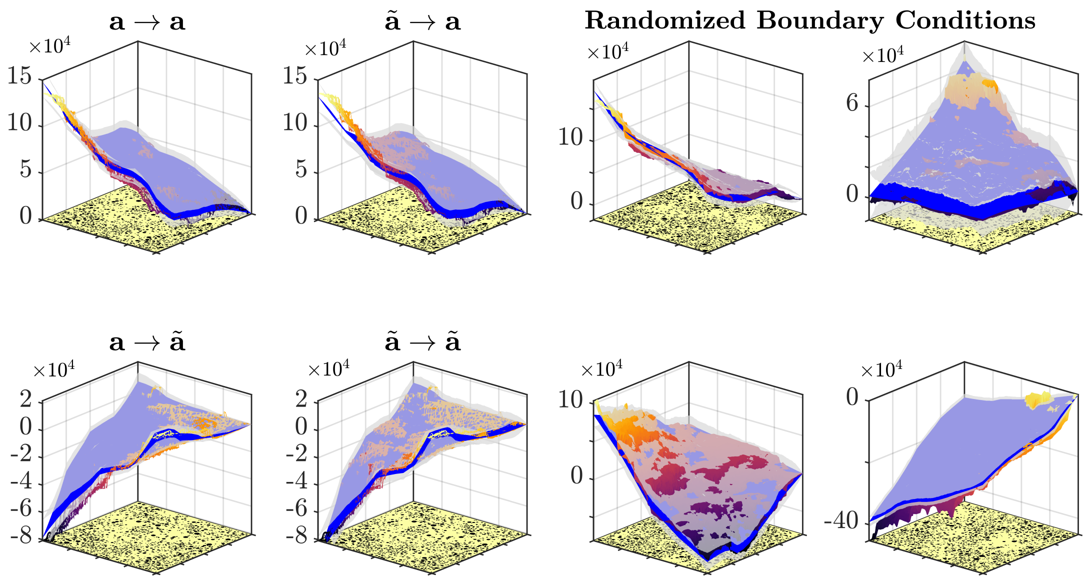


FIGURE 6.15: Left half: Indicative predictive test samples with boundary conditions $\boldsymbol{a}, \tilde{\boldsymbol{a}}$. The model is trained on $N = 128$ training samples on both boundary conditions $\boldsymbol{a}, \tilde{\boldsymbol{a}}$ and predictions are carried out on all training/prediction boundary condition combinations. The deterioration in predictive quality is only slightly visible. Picture adapted from [87].

data and p_{pred} is defined in Section 5.4. The distribution $p(f)$ can then approximated by the posterior

$$p(f) \approx p(f|\mathcal{D}) = \int \delta(f - f(\boldsymbol{u}_f))p_{\text{pred}}(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \mathcal{D})p(\boldsymbol{\lambda}_f)d\boldsymbol{\lambda}_f d\boldsymbol{u}_f. \tag{6.16}$$

As the QoI, we use the FGM pressure response $P(\boldsymbol{x}_0)$ at the upper right corner of the domain Ω, $\boldsymbol{x}_0 = (1 \;\; 1)^T$. The result is shown in Figure 6.16, where the blue line depicts a kernel density estimate of a Monte Carlo based histogram with 10^4 FGM evaluations as an approximation to $p(f)$. The black line shows $p(f|\mathcal{D})$ as defined in Equation (6.16), where an $N = 32$ training set and CGM with permeability field discretization $\dim(\boldsymbol{\lambda}_c) = 4 \times 4$ is used. The 90% credible intervals visualized by the gray shaded areas are a consequence of parameter uncertainty due to limited training data and are obtained by sampling model parameters $\tilde{\boldsymbol{\theta}}_c^{(s)}$, $\boldsymbol{\tau}_c^{(s)}$ and $\boldsymbol{\tau}_{cf}^{(s)}$ from their (approximate) posteriors $Q_{\tilde{\boldsymbol{\theta}}_c}$, $Q_{\boldsymbol{\tau}_c}$ and $Q_{\boldsymbol{\tau}_{cf}}$ to yield estimates of

$$p(f|\tilde{\boldsymbol{\theta}}_c^{(s)}, \boldsymbol{\tau}_c^{(s)}, \boldsymbol{\tau}_{cf}^{(s)}, \mathcal{D}) = \int \delta(f - f(\boldsymbol{u}_f))p_{\text{pred}}(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \tilde{\boldsymbol{\theta}}_c^{(s)}, \boldsymbol{\tau}_c^{(s)}, \boldsymbol{\tau}_{cf}^{(s)}, \mathcal{D})p(\boldsymbol{\lambda}_f)d\boldsymbol{\lambda}_f d\boldsymbol{u}_f. \tag{6.17}$$

The mean and variance due to different model parameters $\tilde{\boldsymbol{\theta}}_c^{(s)}$, $\boldsymbol{\tau}_c^{(s)}$ and $\boldsymbol{\tau}_{cf}^{(s)}$ of every bin of the histogram are then used to compute the 90% credible intervals depicted in Figure 6.16. As can be seen in the figure, the true, Monte Carlo based estimate of $p(f)$ is contained in the credible intervals for its most part.

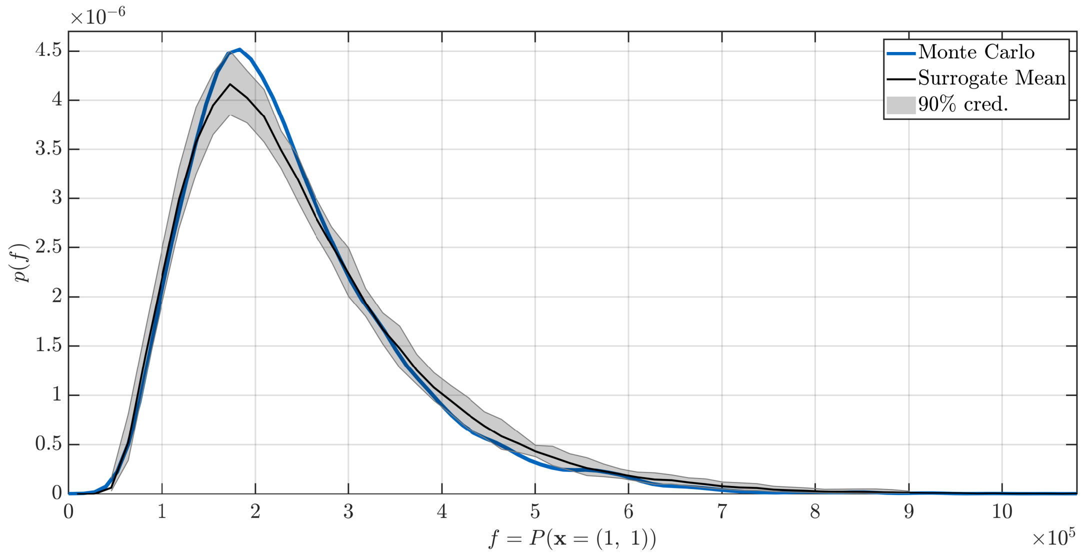


FIGURE 6.16: Probability density function $p(f)$ (blue) of the FGM pressure $P(\boldsymbol{x}_0)$ at $\boldsymbol{x}_0 = (1\ \ 1)^T$ obtained with a kernel density estimate based on 10,000 Monte Carlo samples. The black line depicts $p(f|\mathcal{D})$ as defined by Equation (6.16) and is based on a surrogate with $N = 32$ training samples and a CGM with $\dim(\boldsymbol{\lambda}_c) = 4 \times 4$ permeability field discretization. The gray shaded uncertainty bounds are obtained by first computing separate histograms for $p(f|\tilde{\boldsymbol{\theta}}_c^{(s)}, \boldsymbol{\tau}_c^{(s)}, \boldsymbol{\tau}_{cf}^{(s)}, \mathcal{D})$, and then using the variance of each bin to compute the 90% credible intervals assuming a Gaussian distribution. Picture taken from [87].

6.2.10 Adaptive coarse model refinement

This section gives an experimental illustration of the adaptive CGM permeability field refinement procedure suggested in Section 5.6. To show the potential of the proposed scheme, the aim is to find, starting from a regular $\dim(\boldsymbol{\lambda}_c) = 2 \times 2$ square grid CGM permeability field and performing 4 cell splits, a $\dim(\boldsymbol{\lambda}_c) = 16$-dimensional CGM permeability field discretization that has a higher log evidence lower bound (ELBO) than one with a regular 4×4 square grid.

As the following example should only give a proof of concept, we design material microstructures in such a way that refinement in a certain area of the domain Ω is encouraged. We do so in drawing the microstructures $\boldsymbol{\lambda}_f$ from a distribution generating what we call “tiled” microstructures, which have constant density of circular exclusions all over the unit square domain $\Omega = [0, 1]^2$ except for the lower left quarter $\Omega_{ll} = [0, 0.5]^2$, where it is assumed that the density of exclusions is constant in sub-cells of size 0.125×0.125, but different from sub-cell to sub-cell as depicted in Figure 6.17. Due to this designed inhomogeneity of the material microstructures, it is expected that it is beneficial for the CGM permeability field to be refined in the lower left sub-cell $\Omega_{ll} = [0.5, 0.5]^2$. To reduce the influence of boundary conditions on the cell splitting procedure, boundary conditions of the form $a_x = 1, a_y = a_{xy} = 0$ are applied such that the expected flow velocity is $\langle \boldsymbol{V} \rangle = (1\ \ 0)^T$ all over the domain, i.e., approximately the same fluxes in all cells of same size.

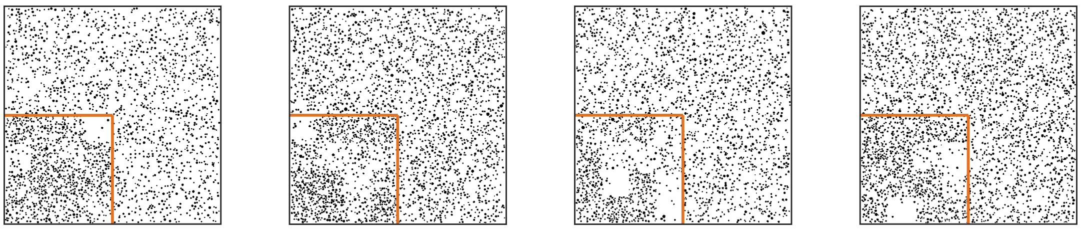


FIGURE 6.17: Four representative samples of so-called "tiled" microstructures, where the density of spherical exclusions (black dots) corresponding to impermeable solid material is distributed homogeneously over the unit square domain Ω except for the lower left quarter Ω_{ll} (indicated by the orange lines). There, the density of exclusions is constant in sub-cells of 0.125×0.125 size, but varies from sub-cell to sub-cell. Such microstructures enforce refinement in the lower left quarter Ω_{ll}. Picture taken from [87].

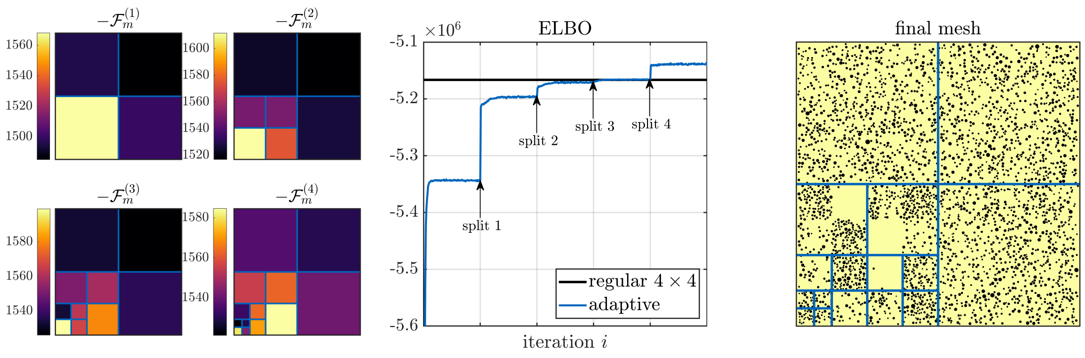


FIGURE 6.18: Left: The negative cell-scoring function $-\mathcal{F}_m^{(i)}$ right before each split i (brighter color stands for lower $\mathcal{F}_m$). Middle: log evidence lower bound (ELBO) as given in Equation (5.89) for the adaptively refined model described in the text (blue line). The black line indicates the ELBO of a model with 4×4 square grid CGM permeability field discretization for comparison. Right: Final mesh after four cell splits (blue lines) with a representative "tiled" microstructure underneath. Picture taken from [87].

The model training and adaptive refinement procedure is carried out on an $N = 32$ training dataset starting with a $\dim(\boldsymbol{\lambda}_c) = 2 \times 2$ CGM. After training convergence, the cell scoring function $\mathcal{F}_m^{(1)}$ established in Equation (5.91) is evaluated (see top left of Figure 6.18). The cell with lowest $\mathcal{F}_m^{(1)}$ (or highest $-\mathcal{F}_m^{(1)}$, as plotted) is chosen and split into four equally sized sub-cells, i.e., the latent space dimension $\dim(\boldsymbol{\lambda}_c) = 4$ is increasing to $\dim(\boldsymbol{\lambda}_c) = 4 - 1 + 4 = 7$. After splitting, the model training continues with the new CGM permeability field discretization and all model parameters initialized to values deduced from the ones found before the cell was split. Training is again run to convergence and the previously stalled ELBO depicted in the middle plot of Figure 6.18 can increase further. After convergence, the next cell with lowest $\mathcal{F}_m^{(2)}$ is split and so on, until four cells are split and $\dim(\boldsymbol{\lambda}_c) = 16$.

As can be seen in the final mesh on the right part of Figure 6.18, it is indeed the lower left quarter Ω_{ll} which becomes refined by the cell-scoring function $\mathcal{F}_m$. The trend of refinement close to the origin $\boldsymbol{x} = 0$ in the lower left corner and along the lower boundary $x_2 = 0$ can be explained by the traction/pressure boundary condition specified on $\Gamma_P = \{\boldsymbol{x} = 0\}$ only and the fact that circular solid phase exclusions have a minimal distance of 0.003 to the domain boundary, i.e., there is always a narrow "free channel" along the boundary where fluid can flow, and as the boundary conditions are chosen such that the fluid is flowing from left to right, it is beneficial to refine along the lower boundary.

The crucial result of the experiment is that the final value of the ELBO after four splittings and a latent space dimension $\dim(\boldsymbol{\lambda}_c) = 16$ is higher than it is for a regular 4×4 CGM permeability field discretization. This means that the adaptive refinement leads to an improved final model compared to standard square grid discretization.

6.3 Chapter summary

The present chapter provides numerical evidence that the physics-aware surrogate modeling framework for the solution of stochastic PDEs in the context of random heterogeneous media as introduced in Chapter 5 yields cheap and accurate probabilistic predictions to the underlying expensive FGM simulation.

Surrogate modeling frameworks based on CGMs with identical as well as simplified constitutive laws compared to the FGM simulation data are tested in combination with the different prior models suggested in Chapter 5. In particular, the proposed coarse-graining surrogate models are validated (i.e., tested for correctness) by application to fine scale microstructural data in a regime where either the effective material property (i.e., the permeability field $\boldsymbol{K}$) is given in closed form or the CGM already represents an accurate description of the given flow problem (homogenization limit).

After successful validation, the model predictive performance is evaluated in terms of the performance metrics presented in 5.4.1 in dependence of the number of training data N and the CGM resolution/latent space dimension $\dim(\boldsymbol{\lambda}_c)$. This is the perhaps most significant experiment since apart from computational efficiency, the predictive accuracy determines the utility of the proposed model as a surrogate in multi-query applications. Also, not only the accuracy of point estimates (e.g., the predictive mean $\boldsymbol{\mu}_{\text{pred}}$) is evaluated, but also the correctness of the predictive uncertainty by comparing the predictive standard deviation $\boldsymbol{\sigma}_{\text{pred}}$ to the true L_2-error on a test set as well as performance metrics that evaluate the test data likelihood under the predictive distribution.

Approximate posterior distributions $q_{\boldsymbol{\lambda}_c}(\boldsymbol{\lambda}_c^{(n)})$ over the latent variables $\boldsymbol{\lambda}_c^{(n)}$ are computed. These distributions directly encode the effective material permeability field $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c)$. The mean effective permeability $\langle \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c) \rangle_{q_{\boldsymbol{\lambda}_c}}$ is plotted and visually compared to the material microstructure determined by $\boldsymbol{\lambda}_f$.

The sparse pattern of activated feature functions φ_{jm} is shown and it is observed that the more training data N are used, the more feature functions φ_{jm} are activated.

The proposed surrogate shows off its ability to encode salient physical information and to extrapolate by providing accurate predictions even on test data that are very different from the ones seen in the training phase.

The suggested adaptive CGM refinement scheme is shown with an illustrative example and as a use case, a simple uncertainty propagation problem is carried out and compared to plain Monte Carlo.

Chapter 7

Conclusions and future work

7.1 Potential future research directions

7.1.1 Multi-query applications

Once trained, the physics-aware surrogate model that was developed in this thesis can be used as a black box solver providing reliable, fully probabilistic estimates of expensive fine-grained model (FGM) forward evaluations of SPDEs at a fraction of their original computational cost. This black-box property makes coupling to any multi-query application such as (stochastic) optimization, uncertainty propagation, control-, design-, or inverse problems trivial such that a case study in any of those fields should be straightforward to realize. Since predictions of the proposed surrogate modeling technique are fully probabilistic, usage in a Bayesian optimization setting [57, 282, 283] could be worthwhile (if the CGM itself requires considerable computational resources) and provide further insight to design of experiment questions, i.e., for what inputs $\boldsymbol{\lambda}_f$ to generate FGM training data in order to obtain an optimally accurate surrogate.

7.1.2 Modeling improvements

Various model extensions can be thought of: the dimension reduction/ coarse-graining step going from the microstructural description $\boldsymbol{\lambda}_f$ to the reduced representation $\boldsymbol{\lambda}_c$ could make use of recent advances in computer vision by application of CNNs or deep Gaussian process models (see Section 4.5.1, 4.5.2), superseding the need of hand-crafting microstructural feature functions. It is noted though that in the "small data, high dimensions" regime of this work, such models lack of possibilities to incorporate a priori physical knowledge (which was the original motivation to use hand-crafted feature functions from the random media literature) and there is the need of significant regularization to avoid overfitting. Similarly, more general decoder models p_{cf} can be contemplated that make use of the spatial nature of the response $\boldsymbol{u}_f$ by employing, e.g., a Gaussian process with distance dependent covariance function. Another idea is to construct the model such that information is not only passed through (and perturbed by) the CGM, but also through ‘bypassing’ components just as in highway networks [284, 285], thereby expanding the model information bottleneck and enabling more detailed reconstruction of fine scale output variations.

7.1.3 Coarse-grained model corrections

A slightly more sophisticated model extension might be the use of CGM constitutive laws that are more complex than Darcy's law given in Equation (2.6), such as the Forchheimer model [286, 287] (assuming unit viscosity $\mu = 1$)

$$\bar{\boldsymbol{V}} = -\boldsymbol{K}\nabla_{\boldsymbol{x}}\bar{P} + |\bar{\boldsymbol{V}}|\boldsymbol{F}\bar{\boldsymbol{V}}, \tag{7.1}$$

where $\boldsymbol{F}$ is the second order tensor Forchheimer coefficient. Another example is the Brinkman law [288],

$$\bar{\boldsymbol{V}} = -\boldsymbol{K}\nabla_{\boldsymbol{x}}\bar{P} + \eta\Delta_{\boldsymbol{x}}\bar{\boldsymbol{V}}, \tag{7.2}$$

where η is called effective viscosity. Moreover, the Darcy flow equation of motion may be extended by a convective term $\bar{\boldsymbol{V}}_{\text{conv}}(\boldsymbol{x})$

$$\bar{\boldsymbol{V}} = -\boldsymbol{K}\nabla_{\boldsymbol{x}}\bar{P} + \bar{P}\bar{\boldsymbol{V}}_{\text{conv}}. \tag{7.3}$$

Also, a source term s may be included such that, combining all of the above constitutive law corrections, the CG model would be

$$\nabla_{\boldsymbol{x}} \cdot \left(-\boldsymbol{K}(\boldsymbol{\lambda}_c)\nabla_{\boldsymbol{x}}\bar{P} + |\bar{\boldsymbol{V}}|\boldsymbol{F}\bar{\boldsymbol{V}} + \eta\Delta_{\boldsymbol{x}}\bar{\boldsymbol{V}} + \bar{P}\bar{\boldsymbol{V}}_{\text{conv}}\right) = s \tag{7.4}$$

and the CGM output $\boldsymbol{u}_c$ becomes a function of the PDE coefficients and source term, $\boldsymbol{u}_c = \boldsymbol{u}_c(\boldsymbol{\lambda}_c, \boldsymbol{F}, \eta, \bar{\boldsymbol{V}}_{\text{conv}}, s)$. The correction term coefficients should be equipped with a prior and be treated as latent variables such that the likelihood of a single FGM data pair $\{\boldsymbol{\lambda}_f, \boldsymbol{u}_f\}$ is written as

$$\begin{aligned} p(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c, \boldsymbol{\theta}_{cf}, \boldsymbol{\theta}_{\text{correct}}) = \int & p_{cf}(\boldsymbol{u}_f|\boldsymbol{u}_c(\boldsymbol{\lambda}_c, \boldsymbol{F}, \eta, \bar{\boldsymbol{V}}_{\text{conv}}, s), \boldsymbol{\theta}_{cf})p_c(\boldsymbol{\lambda}_c|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_c) \\ & p_{\text{correct}}(\boldsymbol{F}, \eta, \bar{\boldsymbol{V}}_{\text{conv}}, s|\boldsymbol{\theta}_{\text{correct}})d\boldsymbol{\lambda}_c\, d\boldsymbol{F}\, d\eta\, d\bar{\boldsymbol{V}}_{\text{conv}}\, ds. \end{aligned} \tag{7.5}$$

The prior over the correction coefficients $p_{\text{correct}}(\boldsymbol{F}, \eta, \bar{\boldsymbol{V}}_{\text{conv}}, s|\boldsymbol{\theta}_{\text{correct}})$ should be sparsity enforcing, such that the model automatically picks the constitutive law that is in best accordance with the data, while setting as many correction terms as possible to zero. One may start off with spatially constant corrections and later generalize to $\boldsymbol{F} = \boldsymbol{F}(\boldsymbol{x}), \eta = \eta(\boldsymbol{x}), \bar{\boldsymbol{V}}_{\text{conv}} = \bar{\boldsymbol{V}}_{\text{conv}}(\boldsymbol{x})$ with constant values in every finite element. Also, one could try to absorb microstructural variability in the correction terms by including dependence on $\boldsymbol{\lambda}_f$, $p_{\text{correct}} = p_{\text{correct}}(\boldsymbol{F}, \eta, \bar{\boldsymbol{V}}_{\text{conv}}, s|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_{\text{correct}})$ using, e.g., a similar feature function based model as for the permeability $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c)$. A surrogate model using a CGM with correction terms as described above that are turned off and on by a sparsity enforcing prior can be particularly interesting for FGM data given by "real-world snapshots", e.g., CT scans of microstructures and pressure field measurements where the constitutive behavior of the FGM flow is unknown but could be revealed by the surrogate model using sparsity priors on the correction terms as described above.

7.1.4 Extension to semi-supervised training data

A frequent machine learning setting is where there are only few labeled training data available, but many unlabeled input samples exist, i.e., $\mathcal{D}_{\text{sup}} = \mathcal{D} = \left\{\boldsymbol{\lambda}_f^{(n)}, \boldsymbol{u}_f^{(n)}\right\}_{n=1}^{N}$, $\mathcal{D}_{\text{unsup}} = \left\{\boldsymbol{\lambda}_f^{(n)}\right\}_{n=N+1}^{N+M}$, where N, M are the number of labeled and unlabeled training samples (with usually $M \ll N$), respectively.

To make use of the unsupervised data $\mathcal{D}_{\text{unsup}}$ as well, one may target a *semi-supervised model* [289–291] where both the fine scale microstructures $\boldsymbol{\lambda}_f$ and their reduced representations $\boldsymbol{\lambda}_c$ are generated from a low-dimensional latent space $\mathcal{Z}$ similar to probabilistic PCA [292, 293] like

$$\lambda_{c,m} = g_m(\boldsymbol{z}, \tilde{\boldsymbol{\theta}}_c) + \tau_{c,m}^{-1/2}\xi_m \qquad \xi_m \sim \mathcal{N}(0,1) \tag{7.6}$$

$$\lambda_{f,i} = h_i(\boldsymbol{z}, \tilde{\boldsymbol{\theta}}_f) + \tau_{f,i}^{-1/2}\zeta_i \qquad \zeta_i \sim \mathcal{N}(0,1) \tag{7.7}$$

where $\boldsymbol{z} \in \mathcal{Z}$, $\boldsymbol{g}(\boldsymbol{z}, \tilde{\boldsymbol{\theta}}_c), \boldsymbol{h}(\boldsymbol{z}, \tilde{\boldsymbol{\theta}}_f)$ are arbitrary mapping functions parametrized by $\tilde{\boldsymbol{\theta}}_c, \tilde{\boldsymbol{\theta}}_f$, respectively and $\boldsymbol{\tau}_c, \boldsymbol{\tau}_f$ are noise precision parameters. By convention, it is assumed that the latent variables $\boldsymbol{z}$ are standard normal distributed, $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$. Moreover, it is noted that the reduced representation/effective material property encoding $\boldsymbol{\lambda}_c$ no longer explicitly depends on the microstructure $\boldsymbol{\lambda}_f$, but implicitly via the latent variables $\boldsymbol{z}$. Also observe that $\boldsymbol{\lambda}_c, \boldsymbol{\lambda}_f$ are conditionally independent, i.e., $p(\boldsymbol{\lambda}_c, \boldsymbol{\lambda}_f|\boldsymbol{z}) = p_c(\boldsymbol{\lambda}_c|\boldsymbol{z})p_f(\boldsymbol{\lambda}_f|\boldsymbol{z})$.

To take into account the input microstructures $\boldsymbol{\lambda}_f$ for training, we first switch to a *generative*[1] form for the likelihood function for the *supervised* part of the data $\mathcal{L}_{\text{sup}}(\mathcal{D}_{\text{sup}}|\boldsymbol{\theta}_{cf}, \boldsymbol{\theta}_c, \boldsymbol{\theta}_f)$,

$$\begin{aligned}\mathcal{L}_{\text{sup}}(\mathcal{D}_{\text{sup}}|\boldsymbol{\theta}_{cf}, \boldsymbol{\theta}_c, \boldsymbol{\theta}_f) &= \prod_{n=1}^{N} p(\boldsymbol{u}_f^{(n)}, \boldsymbol{\lambda}_f^{(n)}|\boldsymbol{\theta}_{cf}, \boldsymbol{\theta}_c, \boldsymbol{\theta}_f) \\ = \prod_{n=1}^{N} \int p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\theta}_{cf})&p_c(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{z}^{(n)}, \boldsymbol{\theta}_c)d\boldsymbol{\lambda}_c^{(n)}\cdot \\ &\cdot p_f(\boldsymbol{\lambda}_f^{(n)}|\boldsymbol{z}^{(n)}, \boldsymbol{\theta}_f)p(\boldsymbol{z}^{(n)})d\boldsymbol{z}^{(n)},\end{aligned} \tag{7.8}$$

where $\boldsymbol{\theta}_f = \{\tilde{\boldsymbol{\theta}}_f, \boldsymbol{\tau}_f\}$ denotes the parameters of the input data generative process.

[1]Note the difference to the *discriminative* form of the likelihood function as given in Equation (5.25).

The likelihood for the unsupervised part of the data $\mathcal{D}_{\text{unsup}}$ is readily deduced from the generative process and is given by

$$\mathcal{L}_{\text{unsup}}(\mathcal{D}_{\text{unsup}}|\boldsymbol{\theta}_f) = \prod_{n=N+1}^{N+M} \int p_f(\boldsymbol{\lambda}_f^{(n)}|\boldsymbol{z}^{(n)}, \boldsymbol{\theta}_f) p(\boldsymbol{z}^{(n)}) d\boldsymbol{z}^{(n)}. \tag{7.9}$$

The full data log likelihood is therefore

$$\begin{aligned} &\log \mathcal{L}(\mathcal{D}_{\text{sup}}, \mathcal{D}_{\text{unsup}}|\boldsymbol{\theta}_{cf}, \boldsymbol{\theta}_c, \boldsymbol{\theta}_f) = \\ &= \sum_{n=1}^{N} \log \int p_{cf}(\boldsymbol{u}_f^{(n)}|\boldsymbol{u}_c(\boldsymbol{\lambda}_c^{(n)}), \boldsymbol{\theta}_{cf}) p_c(\boldsymbol{\lambda}_c^{(n)}|\boldsymbol{z}^{(n)}, \boldsymbol{\theta}_c) d\boldsymbol{\lambda}_c^{(n)} \cdot \\ &\qquad \cdot p_f(\boldsymbol{\lambda}_f^{(n)}|\boldsymbol{z}^{(n)}, \boldsymbol{\theta}_f) p(\boldsymbol{z}^{(n)}) d\boldsymbol{z}^{(n)} \\ &+ \sum_{n=N+1}^{N+M} \log \int p_f(\boldsymbol{\lambda}_f^{(n)}|\boldsymbol{z}^{(n)}, \boldsymbol{\theta}_f) p(\boldsymbol{z}^{(n)}) d\boldsymbol{z}^{(n)}. \end{aligned} \tag{7.10}$$

Any of the model parameters $\boldsymbol{\theta}_{cf}, \boldsymbol{\theta}_c, \boldsymbol{\theta}_f$ may again be equipped with any reasonable prior as in Chapter 5 and the model can be trained by maximizing the likelihood function/the posterior/the evidence as before.

Predictive samples can be generated using

$$p(\boldsymbol{u}_f|\boldsymbol{\lambda}_f, \boldsymbol{\theta}_{cf}, \boldsymbol{\theta}_c, \boldsymbol{\theta}_f) = \int p_{cf}(\boldsymbol{u}_f|\boldsymbol{u}_c(\boldsymbol{\lambda}_c), \boldsymbol{\theta}_{cf}) p_c(\boldsymbol{\lambda}_c|\boldsymbol{z}, \boldsymbol{\theta}_c) d\boldsymbol{\lambda}_c p(\boldsymbol{z}|\boldsymbol{\lambda}_f) d\boldsymbol{z}, \tag{7.11}$$

where intricacies may arise since inference w.r.t.

$$p(\boldsymbol{z}|\boldsymbol{\lambda}_f) \propto \mathcal{N}(\boldsymbol{\lambda}_f|\boldsymbol{h}(\boldsymbol{z}, \boldsymbol{\theta}_f), \text{diag}(\boldsymbol{\tau}_f)^{-1}) \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I}) \tag{7.12}$$

is typically not feasible in closed form for nonlinear mappings $h(\boldsymbol{z}, \boldsymbol{\theta}_f)$.

7.2 Conclusions and summary

This thesis has successfully developed a new class of physics-aware, fully probabilistic machine learning models for surrogate modeling of expensive finite element simulations of stochastic partial differential equations in the context of flow problems through random heterogeneous media. By making use of an encoder-decoder model architecture with an effective physics solver (called the *coarse grained model* or CGM) based on much coarser spatial discretizations and potentially simplified governing equations as the core unit, it was shown that accurate, fully probabilistic predictions can be obtained using only $N \approx 10 \ldots 100$ fine grained model (FGM) evaluations as training samples, despite the high-dimensional input uncertainties $\boldsymbol{\lambda}_f, \dim(\boldsymbol{\lambda}_f) \gtrsim 10{,}000$ describing the microstructure of the random medium. The width of the predictive distributions obtained accurately quantified uncertainty due to limited training data and information loss during the coarse-graining process (i.e., due to finite model complexity). A conspicuous feature of the proposed surrogate is the ability to deterministically assign boundary conditions, which enables accurate predictions even under extrapolative conditions. A fully Bayesian framework free of user-specified parameters was developed allowing for simple usage of the model. As a by-product, the model evidence can be estimated and used for model comparison. We presented a method using the evidence to adaptively refine the CGM, thereby increasing model complexity.

Chapter 2 gave an introduction to the physics of fluid flow through random porous media which is the sample application of the surrogate modeling framework that was developed in this work. Starting from Navier-Stokes equations, the governing equations of Stokes flow (also known as creeping flow) were derived under the assumption of low Reynolds numbers $\mathrm{Re} \ll 1$ which is the common scenario for fluid flow through random porous media. Furthermore, the phenomenological law of Darcy flow was presented and its equivalence to Stokes flow in the homogenization limit was outlined, the central building block for the construction of a CGM based on simplified constitutive equations. Moreover, it was shown how to generate realistic random microstructures with specific lower order statistics that can be used for numerical experiments.

Chapter 3 briefly discussed the notion of stochastic partial differential equations (SPDEs), their solution with vanilla Monte Carlo and how surrogate models can be used to accelerate and improve inference. Moreover, the numerical solution of PDEs with the finite element method (FEM) was presented using the example of Stokes and Darcy flow.

In Chapter 4, a broad introduction to relevant topics of uncertainty quantification (UQ), machine learning, and surrogate modeling was given. After stating the uncertainty propagation problem in Section 4.1, the concept of (supervised) machine learning was introduced in Section 4.2 starting from Bayes' law and linear regression. Sparsity enforcing priors that were used in different setups of the proposed physics-aware surrogate modeling framework were introduced and their benefits and disadvantages were pointed out. The model evidence was presented as a means to compare models of different architecture and complexity, which was needed in a later section in the context of adaptive CGM refinement. All approximate inference techniques that were used during this work were presented in Section 4.3, from crude Laplace approximation to modern stochastic variational inference (SVI) methods. A presentation of some of the most popular stochastic gradient ascent (SGA) algorithms needed for SVI was given in Section 4.4. The chapter concluded with a summary of the most popular surrogate modeling techniques for the solution of SPDEs in 4.5, compared their benefits and disadvantages and thereby motivated the need for a physics-aware surrogate that can handle high dimensional stochastic inputs even with small training data.

The development of such a surrogate model is the main contribution of this thesis and its constructive form was outlined in full detail in Chapter 5. We suggested a three-component model where the first part consists of a coarse-graining distribution p_c that is capable to extract the salient microstructural features that contain a maximum amount of information required to reconstruct the PDE response denoted by $\boldsymbol{u}_f$. This information is decoded and compressed by a large library of microstructural feature functions into a low-dimensional, latent representation $\boldsymbol{\lambda}_c$ which is in one-to-one correspondence to an effective material property field (here: permeability) $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{\lambda}_c)$. The effective material property field $\boldsymbol{K}$ then serves as the input to a much coarser discretized CGM based on Darcy's equation of motion which forms the second model component. Thirdly and finally, The output $\boldsymbol{u}_c$ of the CGM, the Darcy flow pressure field, is probabilistically reconstructed to its fine scale counterpart $\boldsymbol{u}_f$ using a decoder distribution p_{cf}. Different sparsity-enforcing prior models were developed for the latent variable model based on well-established machine learning techniques and efficient training algorithms were presented in Section 5.3. An efficient algorithm to generate predictive samples and to compute low order predictive statistical moments was outlined and valuable predictive performance measures were presented in Section 5.4. The full numerical scaling behavior of both training and prediction stages was discussed in Section 5.5. A method for automated adaptive refinement of the CGM core unit was presented in Section 5.6.

Various numerical tests of the proposed surrogate model were conducted and described in Chapter 6. After successful realization of validation experiments, the predictive performance was tested in dependence of the number of training data N and the CGM resolution/bottleneck dimension $\dim(\boldsymbol{\lambda}_c)$. Effective material property fields were visualized and compared to the true porous microstructure. The sparse set of activated microstructural features that the model identified to be most relevant for reconstruction of the FGM response was shown and interpreted for different classes of microstructures. The extrapolative capabilities of the model have been proven by using test data of significantly different (or even random) boundary conditions than in the training set. As a use case of the developed model, a simple uncertainty propagation problem was carried out and model predictions were compared to plain Monte Carlo results. Finally, an indicative example gave a proof of concept for the suggested adaptive CGM refinement strategy.

Appendix A

Feature functions used for the $2D$ Darcy FGM data experiments of Section 6.1.2

Feature functions φ

Index j	Function φ_j	Comment
1	constant	$\varphi_j = 1$
2	SCA	$\varphi_j = \frac{\alpha + \sqrt{\alpha^2 + 4\lambda_{hi}\lambda_{lo}}}{2}$, $\alpha = \lambda_{lo}(2v_{lo} - 1) + \lambda_{hi}(2v_{hi} - 1)$
3–4	Maxwell-Garnett	$\varphi_j = \frac{\lambda_{\text{mat}}}{1 - 2v_{\text{inc}}}$
5–6	Differential Effective-Medium	$\left(\frac{\lambda_{\text{inc}} - \varphi_j}{\lambda_{\text{inc}} - \lambda_{\text{mat}}}\right)\left(\frac{\lambda_{\text{mat}}}{\varphi_j}\right)^{1/2} = 1 - v_{\text{inc}}$
7–12	Lineal path	Lineal path function for certain phase/distance
13–16	Lin. path parameters	a, b parameters of $a \cdot e^{-b \cdot d}$ fit to lineal path
17–18	Number of distinct high/low conducting blobs	
19–22	Number of high/low conducting pixels to cross from left to right/up to down	
23–26	Max. extent of high/low conducting blob in x/y–direction	
27–31	Generalized mean	$\left(\frac{1}{M}\sum_{m=1}^{M}(\lambda_{f,m}^{[k]})^q\right)^{1/q}$
32–37	Max./mean/variance of convex area of high/low conducting blobs	
38–41	Inv. distance of connected path through high/low cond. phase in x/y-direction, 0 if no connected path existent	
42–43	Specific surface	$-4\frac{\partial}{\partial d} S_2(d)\rvert_{d=0}$, with 2-point correlation $S_2(r)$
44–48	"Gaussian linear filter"	compute $w_i = \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_{\text{center}}, a\boldsymbol{I})$ where $\boldsymbol{\mu}_{\text{center}}$ is the macro-element center and x_i are fine-scale element locations. Compute $\varphi_j = \boldsymbol{w}^T \boldsymbol{\lambda}_f^{[k]}$
49	Standard deviation	$\varphi_j = \left\langle (\lambda_{f,i} - \left\langle \lambda_{f,i} \right\rangle)^2 \right\rangle$
50	Log standard deviation	$\varphi_j = \log(\left\langle (\lambda_{f,i} - \left\langle \lambda_{f,i} \right\rangle)^2 \right\rangle)$
51	Ising energy	Energy of a $2d$ Ising system with coupling $J = 1$ and no external field
52–63	Two-point correlations	$\varphi_j = \frac{1}{N_{el,f}^{[k]}} \sum_{i=1}^{N_{el,f}^{[k]}} \mathbb{1}_0(\lambda_{f,i}^{[k]} - \lambda_{f,i+d}^{[k]})$
64–81	Distance transformations	Mean/variance/maximum of distance transforms under different distance metrics
82–88	Local PCA loadings	Perform PCA using every macro-cell $\boldsymbol{\lambda}_f^{[k]}$. Compute projections onto loadings $\varphi_j = \boldsymbol{w}^T \boldsymbol{\lambda}_f^{[k]}$
89–92	Max. extent of high/low conducting blob in x/y–direction of whole microstructure $\boldsymbol{\lambda}_f$	
93–97	SCA, Maxwell-Garnett, Differential Effective Medium on whole microstructure $\boldsymbol{\lambda}_f$	
98–100	Global PCA loadings	Perform PCA using whole microstructures $\boldsymbol{\lambda}_f$. Compute projections onto loadings $\varphi_j = \boldsymbol{w}^T \boldsymbol{\lambda}_f$

TABLE A.1: Set of 100 feature functions φ applied in the $2D$ Darcy flow examples of Section 6.1.2. Table taken from [104].

Table A.1 shows a list of the 100 feature functions used in the $2d$ numerical examples of Section 6.1.2. Features 1–88 take the subset $\boldsymbol{\lambda}_f^{[m]}$ belonging to subregion Ω_m as input, features 89–100 use the whole vector $\boldsymbol{\lambda}_f$.

Appendix B

Feature functions used for the Stokes FGM data experiments of Section 6.2

Table A.1 shows a list of the 150 feature functions used in the numerical examples of Section 6.2. Features 1–55 take the whole microstructure $\boldsymbol{\lambda}_f$ as input, features 56–150 use the subset $\boldsymbol{\lambda}_f^{(m)}$ pertaining to subdomain/cell Ω_m for which $\boldsymbol{K}_m(\boldsymbol{x}, \boldsymbol{\lambda}_c) = e^{\lambda_{c,m}}\boldsymbol{I}$.

Feature functions φ

Index j	**Function** φ_{jm}	**Comment**
1	constant	$\varphi_j = 1$
2	pore fraction in Ω	pore fraction evaluated on full domain Ω
3	log pore fraction in Ω	
4–7	(pore fraction)$^{0.5 \ldots 0.5 \ldots 2.5}$ in Ω	
8	exp(pore fraction)	
9	log SCA in Ω	log self-consist. approximation [1], sec. 18.1.2, inf. contrast limit
10	Maxwell-Approximation in Ω	inf. contrast limit, see [1], sec. 18.2.1
11	log Maxwell-Approximation	
12–17	log chord length density in Ω, $d = 0.05 \ldots 0$	see [1] sec. 6.2.4
18	interface area in Ω	
19	log interface area in Ω	
20–27	$\lvert$ log interface area$\rvert^{3/2,1/2,1/3,1/4,1/5,2,3,4}$ in Ω	
28–31	(interface area)$^{1/3,1/4,1/5,2}$ in Ω	
32	mean distance edge in Ω	measured from excl. edge to edge
33	log mean distance edge in Ω	measured from excl. edge to edge
34	$\log^2$ mean distance edge in Ω	measured from excl. edge to edge
35	$\log^3$ mean distance edge in Ω	measured from excl. edge to edge
36	mean distance center in Ω	measured from excl. center to center
37	min. distance center in Ω	measured from excl. center to center
38	log min. distance center in Ω	measured from excl. center to center
39	$\log^2$ min. distance center in Ω	measured from excl. center to center
40–43	lineal path in Ω for $d = 0.025, 0.01, 0.005, 0.002$	see [1], sec. 2.4
44–47	log lineal path in Ω for $d = 0.025, 0.01, 0.005, 0.002$	
48	void nearest-neighbor pdf, $d = 0$, in Ω	see [1], sec. 2.8
49	log void nearest-neighbor pdf, $d = 0$, in Ω	see [1]. sec. 2.8
50	pore size density, $d = 0$, in Ω	see [1], sec. 2.6
51	log pore size density, $d = 0$, in Ω	see [1], sec. 2.6
52	mean chord length in Ω	see [1], sec. 2.5
53	log mean chord length in Ω	see [1], sec. 2.5
54	exp mean chord length in Ω	see [1], sec. 2.5
55	(mean chord length)$^{0.5}$ in Ω	see [1], sec. 2.5
56–58	$\left\langle r_{ex}^{0.2,0.5,1} \right\rangle$ in Ω_m	expected exclusion radii moments
59–61	$\log \left\langle r_{ex}^{0.2,0.5,1} \right\rangle$ in Ω_m	log expected exclusion radii moments
62	pore fraction in Ω_m	
63	log pore fraction in Ω_m	
64	exp pore fraction in Ω_m	
65–68	(pore fraction)$^{0.5,1.5,2,2.5}$ in Ω_m	
69	log self-consistent approximation (inf. contrast) in Ω_m	
70	Maxwell Approximation in Ω_m	
71	log Maxwell Approximation in Ω_m	
72–78	log chord length dens. in Ω_m, $d = (50, 25, 12.5, 6.25, 3, 1.5, 0) \cdot 10^{-4}$	
79	interface area in Ω_m	
80–88	$\lvert$ log interface area$\rvert^{1,1.5,2,3,4,5,1/2,1/3,1/4}$ in Ω_m	
89–93	$\lvert$ interface area$\rvert^{1/2,1/3,1/4,1/5,2}$ in Ω_m	
94	mean distance edge in Ω_m	measured from excl. edge to edge
95	log mean distance edge in Ω_m	measured from excl. edge to edge
96	max distance edge in Ω_m	measured from excl. edge to edge
97	log max distance edge in Ω_m	measured from excl. edge to edge
98	std distance edge in Ω_m	measured from excl. edge to edge
99	log std distance edge in Ω_m	measured from excl. edge to edge
100–104	square well potential, width = $(1, 2, 3, 4, 5) \cdot 10^{-2}$ in Ω_m	
105	$\log^2$ mean distance in Ω_m	measured from excl. edge to edge
106	$\log^3$ mean distance in Ω_m	measured from excl. edge to edge
107	mean distance center in Ω_m	measured from excl. center to center
108	min. distance center in Ω_m	measured from excl. center to center
109	log min. distance center in Ω_m	measured from excl. center to center
110	$\log^2$ min. distance center in Ω_m	measured from excl. center to center
111–114	lineal path in Ω_m for $d = 0.025, 0.01, 0.005, 0.002$	
115–118	log lineal path in Ω_m for $d = 0.025, 0.01, 0.005, 0.002$	
119	void nearest-neighbor pdf, $d = 0$, in Ω_m	see [1], sec. 2.8
120	log void nearest-neighbor pdf, $d = 0$, in Ω_m	see [1]. sec. 2.8
121	pore size density, $d = 0$, in Ω_m	see [1], sec. 2.6
122	log pore size density, $d = 0$, in Ω_m	see [1], sec. 2.6
123	mean chord length in Ω_m	see [1], sec. 2.5
124	log mean chord length in Ω_m	see [1], sec. 2.5
125	exp mean chord length in Ω_m	see [1], sec. 2.5
126	(mean chord length)$^{0.5}$ in Ω_m	see [1], sec. 2.5
127–129	$\left\langle r_{ex}^{0.2,0.5,1} \right\rangle$ in Ω_m	expected exclusion radii moments
130–132	$\log \left\langle r_{ex}^{0.2,0.5,1} \right\rangle$ in Ω_m	log expected exclusion radii moments
133	length scale of exp. approx. to lin. path in Ω_m	
134	mean of euclidean dist. transform in Ω_m	see [93]
135	variance of euclidean dist. transform in Ω_m	see [93]
136	max. of euclidean dist. transform in Ω_m	see [93]
137	mean of chessboard dist. transform in Ω_m	see [93]
138	variance of chessboard dist. transform in Ω_m	see [93]
139	max. of chessboard dist. transform in Ω_m	see [93]
140	mean of cityblock dist. transform in Ω_m	see [93]
141	variance of cityblock dist. transform in Ω_m	see [93]
142	max. of cityblock dist. transform in Ω_m	see [93]
143–145	Gauss lin. filt. $d = 2, 5, 10$ pixels in Ω_m	see [104]
146	Ising energy in Ω_m	
147	shortest connected path, x-dir., Euclidean, in Ω_m	see [93]
148	shortest connected path, y-dir., Euclidean, in Ω_m	see [93]
149	shortest connected path, x-dir., cityblock, in Ω_m	see [93]
150	shortest connected path, y-dir., cityblock, in Ω_m	see [93]

TABLE B.1: Set of 150 feature functions φ applied in the numerical examples of Section 6.2.

List of abbreviations

ADAM	**ADA**ptive **M**oment estimation (SGA algorithm)
ANN	**A**rtificial **N**eural **N**etwork
ARD	**A**utomatic **R**elevance **D**etermination
BP	(Error) **B**ack**P**rpoagation
DNN	**D**eep **N**eural **N**etwork
ELBO	**E**vidence **L**ower **BO**und
EM	**E**xpectation **M**aximization (algorithm)
FE	**F**inite **E**lements
FEM	**F**inite **E**lement **M**ethod
FGM	**F**ine **G**rained **M**odel
FOM	**F**ull **O**rder **M**odel
GP	**G**aussian **P**rocess
gPC	**g**eneralized **P**olynomial **C**haos
HMC	**H**amiltonian (or **H**ybrid) **M**onte **C**arlo
i.i.d.	**i**ndependent and **i**dentically **d**istributed (random variable)
KL	**K**ullback-**L**eibler (divergence)
MALA	**M**etropolis-**A**djusted **L**angevin **A**lgorithm
MAP	**M**aximum **a** **p**osteriori
MCMC	**M**arkov **c**hain **M**onte **C**arlo
MGA	**M**axwell **G**arnett **A**pproximation
MH	**M**etropolis **H**astings (algorithm)
ML	**M**aximum **l**ikelihood
MLL	**M**ean **L**og **L**ikelihood
MLP	**M**ulti**L**ayer **P**erceptron
PDE	**P**artial **D**ifferential **E**quation
POD	**P**roper **O**rthogonal **D**ecomposition
RB	**R**educed **B**asis (method)
ROM	**R**educed **O**rder **M**odel
RVM	**R**epresentative **V**olume **E**lement
SC	**S**tochastic **C**ollocation
SCA	**S**elf - **C**onsistent **A**pproximation (Bruggeman formula)
SGA	**S**tochastic **G**radient **A**scent
SMC	**S**equential **M**onte **C**arlo
SPDE	**S**tochastic **P**artial **D**ifferential **E**quation
s.t.	**s**uch **t**hat

SVD	**S**ingular **V**alue **D**ecomposition
SVI	**S**tochastic **V**ariational **I**nference
UP	**U**ncertainty **P**ropagation
UQ	**U**ncertainty **Q**uantification
VI	**V**ariational **I**nference
VRVM	**V**ariational **R**elevance **V**ector **M**achine
w.r.t.	**w**ith **r**espect **t**o

Bibliography

[1] S. Torquato. *Random Heterogeneous Materials: Microstructure and Macroscopic Properties*. Springer New York, 2002. DOI: 10.1007/978-1-4757-6355-3.

[2] R. Ghanem, D. Higdon, and H. Owhadi. *Handbook of Uncertainty Quantification*. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-12385-1.

[3] T. J. Sullivan. *Introduction to Uncertainty Quantification*. Springer International Publishing, 2015. DOI: 10.1007/978-3-319-23395-6.

[4] C. Soize. *Uncertainty Quantification: An Accelerated Course with Advanced Applications in Computational Engineering*. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-54339-0.

[5] M. Grigoriu. *Stochastic Systems: Uncertainty Quantification and Propagation*. Springer London, 2012. DOI: 10.1007/978-1-4471-2327-9.

[6] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005. DOI: 10.1137/1.9780898717921.

[7] A. Kirsch. *An Introduction to the Mathematical Theory of Inverse Problems*. Springer New York, 2011. DOI: 10.1007/978-1-4419-8474-6.

[8] W. Menke. *Geophysical Data Analysis: Discrete Inverse Theory*. Third Edition. Academic Press, 2012. DOI: 10.1016/C2011-0-69765-0.

[9] M. Bertero and P. Bocacci. *Introduction to Inverse Problems in Imaging*. Taylor & Francis, 1998. DOI: 10.1201/9781439822067.

[10] J. Wang and N. Zabaras. "A Bayesian inference approach to the inverse heat conduction problem". In: *International Journal of Heat and Mass Transfer* 47.17 (2004), pp. 3927 –3941. DOI: 10.1016/j.ijheatmasstransfer.2004.02.028.

[11] I. Bilionis and N. Zabaras. "Solution of inverse problems with limited forward solver evaluations: a Bayesian perspective". In: *Inverse Problems* 30.1 (2013), p. 015004. DOI: 10.1088/0266-5611/30/1/015004.

[12] I. Franck and P. Koutsourelakis. "Multimodal, high-dimensional, model-based, Bayesian inverse problems with applications in biomechanics". In: *Journal of Computational Physics* 329 (2017), pp. 91 –125. DOI: 10.1016/j.jcp.2016.10.039.

[13] I. M. Franck. "Sparse Variational Bayesian algorithms for large-scale inverse problems with applications in biomechanics". Dissertation. Technische Universität München, 2017. DOI: 10.14459/2017md1343317.

[14] R. G. Ghanem and P. D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Springer, 1991. DOI: 10.1007/978-1-4612-3094-6.

[15] D. Xiu and G. E. Karniadakis. "The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations". In: *SIAM Journal on Scientific Computing* 24.2 (2002), pp. 619 –644. DOI: 10.1137/S1064827501387826.

[16] D. Xiu. *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, 2010. DOI: 10.2307/j.ctv7h0skv.

[17] N. Wiener. "The Homogeneous Chaos". In: *American Journal of Mathematics* 60.4 (1938), pp. 897–936. DOI: 10.2307/2371268.

[18] B. Sudret. "Global sensitivity analysis using polynomial chaos expansions". In: *Reliability Engineering & System Safety* 93.7 (2008). Bayesian Networks in Dependability, pp. 964–979. DOI: 10.1016/j.ress.2007.04.002.

[19] T. Crestaux, O. L. Maître, and J.-M. Martinez. "Polynomial chaos expansion for sensitivity analysis". In: *Reliability Engineering & System Safety* 94.7 (2009). Special Issue on Sensitivity Analysis, pp. 1161–1172. DOI: 10.1016/j.ress.2008.10.008.

[20] D. Xiu and G. E. Karniadakis. "Modeling uncertainty in flow simulations via generalized polynomial chaos". In: *Journal of Computational Physics* 187.1 (2003), pp. 137–167. DOI: 10.1016/S0021-9991(03)00092-5.

[21] H. N. Najm. "Uncertainty Quantification and Polynomial Chaos Techniques in Computational Fluid Dynamics". In: *Annual Review of Fluid Mechanics* 41.1 (2009), pp. 35–52. DOI: 10.1146/annurev.fluid.010908.165248.

[22] Y. M. Marzouk, H. N. Najm, and L. A. Rahn. "Stochastic spectral methods for efficient Bayesian solution of inverse problems". In: *Journal of Computational Physics* 224.2 (2007), pp. 560–586. DOI: 10.1016/j.jcp.2006.10.010.

[23] Y. M. Marzouk and H. N. Najm. "Dimensionality reduction and polynomial chaos acceleration of Bayesian inference in inverse problems". In: *Journal of Computational Physics* 228.6 (2009), pp. 1862–1902. DOI: 10.1016/j.jcp.2008.11.024.

[24] I. Babuška, F. Nobile, and R. Tempone. "A Stochastic Collocation Method for Elliptic Partial Differential Equations with Random Input Data". In: *SIAM Journal on Numerical Analysis* 45.3 (2007), pp. 1005–1034. DOI: 10.1137/050645142.

[25] F. Nobile, R. Tempone, and C. Webster. "A Sparse Grid Stochastic Collocation Method for Partial Differential Equations with Random Input Data". In: *SIAM Journal on Numerical Analysis* 46.5 (2008), pp. 2309–2345. DOI: 10.1137/060663660.

[26] X. Ma and N. Zabaras. "An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations". In: *Journal of Computational Physics* 228.8 (2009), pp. 3084 –3113. DOI: 10.1016/j.jcp.2009.01.006.

[27] B. Ganapathysubramanian and N. Zabaras. “Sparse grid collocation schemes for stochastic natural convection problems”. In: *Journal of Computational Physics* 225.1 (2007), pp. 652 –685. DOI: 10.1016/j.jcp.2006.12.014.

[28] D. Zhang and Z. Lu. “An efficient, high-order perturbation approach for flow in random porous media via Karhunen–Loève and polynomial expansions”. In: *Journal of Computational Physics* 194.2 (2004), pp. 773–794. DOI: 10.1016/j.jcp.2003.09.015.

[29] C. W. Rowley, T. Colonius, and R. M. Murray. “Model reduction for compressible flows using POD and Galerkin projection”. In: *Physica D: Nonlinear Phenomena* 189.1 (2004), pp. 115 –129. DOI: 10.1016/j.physd.2003.03.001.

[30] P. Benner, S. Gugercin, and K. Willcox. “A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems”. In: *SIAM Review* 57.4 (2015), pp. 483–531. DOI: 10.1137/130932715.

[31] J. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-22470-1.

[32] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations. An Introduction*. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-15431-2.

[33] G. Rozza, D. B. P. Huynh, and A. T. Patera. “Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations”. In: *Archives of Computational Methods in Engineering* 15.3 (2007). DOI: 10.1007/BF03024948.

[34] G. Rozza et al. “Reduced Basis Methods and A Posteriori Error Estimation for Parametrized Partial Differential Equations”. In: *MIT Pappalardo Graduate Monographs in Mechanical Engineering*. 2007. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.631.2901.

[35] M. Couplet, C. Basdevant, and P. Sagaut. “Calibrated reduced-order POD-Galerkin system for fluid flow modelling”. In: *Journal of Computational Physics* 207.1 (2005), pp. 192–220. DOI: 10.1016/j.jcp.2005.01.008.

[36] A. Quarteroni, G. Rozza, and A. Manzoni. “Certified Reduced Basis Approximation for Parametrized Partial Differential Equations and Applications”. In: *Journal of Mathematics in Industry* 1 (2011). DOI: 10.1186/2190-5983-1-3.

[37] T. Cui, Y. Marzouk, and K. Willcox. “Data-driven model reduction for the Bayesian solution of inverse problems”. In: *International Journal for Numerical Methods in Engineering* 102.5 (2015), pp. 966–990. DOI: 10.1002/nme.4748.

[38] B. Afkham and J. Hesthaven. “Structure Preserving Model Reduction of Parametric Hamiltonian Systems”. In: *SIAM Journal on Scientific Computing* 39.6 (2017), A2616–A2644. DOI: 10.1137/17M1111991.

[39] M. Guo and J. Hesthaven. “Reduced order modeling for nonlinear structural analysis using Gaussian process regression”. In: *Computer Methods in Applied*

Mechanics and Engineering 341 (2018), pp. 807–826. DOI: 10.1016/j.cma.2018.07.017.

[40] J. Hesthaven and S. Ubbiali. "Non-intrusive reduced order modeling of nonlinear problems using neural networks". In: *Journal of Computational Physics* 363 (2018), pp. 55–78. DOI: 10.1016/j.jcp.2018.02.037.

[41] Q. Wang, J. Hesthaven, and D. Ray. "Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem". In: *Journal of Computational Physics* 384 (2019), pp. 289–307. DOI: 10.1016/j.jcp.2019.01.031.

[42] A. Iollo, S. Lanteri, and J.-A. Désidéri. "Stability Properties of POD–Galerkin Approximations for the Compressible Navier–Stokes Equations". In: *Theoretical and Computational Fluid Dynamics* 13.6 (2000), pp. 377–396. DOI: 10.1007/s001620050119.

[43] F. Ballarin et al. "Supremizer stabilization of POD–Galerkin approximation of parametrized steady incompressible Navier–Stokes equations". In: *International Journal for Numerical Methods in Engineering* 102.5 (2015), pp. 1136–1161. DOI: 10.1002/nme.4772.

[44] J.-Y. Zhu et al. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017. DOI: 10.1109/ICCV.2017.244.

[45] P. Isola et al. "Image-to-image translation with conditional adversarial networks". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017. DOI: 10.1109/cvpr.2017.632.

[46] Y. Zhu and N. Zabaras. "Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification". In: *Journal of Computational Physics* 366 (2018), pp. 415–447. DOI: 10.1016/j.jcp.2018.04.018.

[47] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL: http://www.gaussianprocess.org/gpml/.

[48] I. Bilionis and N. Zabaras. "Bayesian Uncertainty Propagation Using Gaussian Processes". In: *Handbook of Uncertainty Quantification*. Springer International Publishing, 2017, pp. 555–599. DOI: 10.1007/978-3-319-12385-1_16.

[49] I. Bilionis et al. "Multi-output separable Gaussian process: Towards an efficient, fully Bayesian paradigm for uncertainty quantification". In: *Journal of Computational Physics* 241 (2013), pp. 212 –239. DOI: 10.1016/j.jcp.2013.01.011.

[50] H.-P. Wan et al. "Analytical uncertainty quantification for modal frequencies with structural parameter uncertainty using a Gaussian process metamodel". In: *Engineering Structures* 75 (2014), pp. 577–589. DOI: 10.1016/j.engstruct.2014.06.028.

[51] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Machine learning of linear differential equations using Gaussian processes”. In: *Journal of Computational Physics* 348 (2017), pp. 683–693. DOI: 10.1016/j.jcp.2017.07.050.

[52] A. O’Hagan and M. C. Kennedy. “Predicting the output from a complex computer code when fast approximations are available”. In: *Biometrika* 87.1 (2000), pp. 1–13. DOI: 10.1093/biomet/87.1.1.

[53] S. Lee et al. “Linking Gaussian process regression with data-driven manifold embeddings for nonlinear data fusion”. In: *Interface Focus* 9.3 (2019). DOI: 10.1098/rsfs.2018.0083.

[54] P. Perdikaris et al. “Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science* 473.2198 (2017). DOI: 10.1098/rspa.2016.0751.

[55] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Inferring solutions of differential equations using noisy multi-fidelity data”. In: *Journal of Computational Physics* 335 (2017), pp. 736–746. DOI: 10.1016/j.jcp.2017.01.060.

[56] P. Perdikaris, D. Venturi, and G. E. Karniadakis. “Multifidelity Information Fusion Algorithms for High-Dimensional Systems and Massive Data sets”. In: *SIAM Journal on Scientific Computing* 38.4 (2016), B521–B538. DOI: 10.1137/15M1055164.

[57] P. Perdikaris and G. E. Karniadakis. “Model inversion via multi-fidelity Bayesian optimization: a new paradigm for parameter estimation in haemodynamics, and beyond”. In: *Journal of The Royal Society Interface* 13.118 (2016). DOI: 10.1098/rsif.2015.1107.

[58] L. Parussini et al. “Multi-fidelity Gaussian process regression for prediction of random fields”. In: *Journal of Computational Physics* 336 (2017), pp. 36–50. DOI: 10.1016/j.jcp.2017.01.047.

[59] P. Perdikaris, L. Grinberg, and G. E. Karniadakis. “Multiscale modeling and simulation of brain blood flow”. In: *Physics of Fluids* 28.2 (2016). DOI: 10.1063/1.4941315.

[60] P. Perdikaris et al. “Multi-fidelity modelling via recursive co-kriging and Gaussian–Markov random fields”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2179 (2015). DOI: 10.1098/rspa.2015.0018.

[61] R. Tripathy, I. Bilionis, and M. Gonzalez. “Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation”. In: *Journal of Computational Physics* 321 (2016), pp. 191–223. DOI: 10.1016/j.jcp.2016.05.039.

[62] M. Raissi and G. E. Karniadakis. “Hidden physics models: Machine learning of nonlinear partial differential equations”. In: *Journal of Computational Physics* 357 (2018), pp. 125–141. DOI: 10.1016/j.jcp.2017.11.039.

[63] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[64] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[65] K. Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological Cybernetics* 36.4 (1980), pp. 193–202. DOI: 10.1007/BF00344251.

[66] C. Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. DOI: 10.1109/CVPR.2016.308.

[67] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1097–1105. DOI: 10.1145/3065386.

[68] I. Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[69] I. Goodfellow. "NIPS 2016 Tutorial: Generative Adversarial Networks". In: *arXiv e-print* (2016). URL: https://arxiv.org/abs/1701.00160.

[70] A. Paszke et al. "Automatic Differentiation in PyTorch". In: *NIPS Autodiff Workshop*. 2017. URL: https://openreview.net/forum?id=BJJsrmfCZ.

[71] M. Abadi et al. "TensorFlow: A System for Large-Scale Machine Learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USENIX Association, 2016, pp. 265–283. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

[72] R. K. Tripathy and I. Bilionis. "Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification". In: *Journal of Computational Physics* 375 (2018), pp. 565–588. DOI: 10.1016/j.jcp.2018.08.036.

[73] Y. Zhu et al. "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data". In: *Journal of Computational Physics* 394 (2019), pp. 56–81. DOI: 10.1016/j.jcp.2019.05.024.

[74] S. Mo et al. "Deep Autoregressive Neural Networks for High-Dimensional Inverse Problems in Groundwater Contaminant Source Identification". In: *Water Resources Research* 55.5 (2019), pp. 3856–3881. DOI: 10.1029/2018WR024638.

[75] S. Mo et al. "Deep Convolutional Encoder-Decoder Networks for Uncertainty Quantification of Dynamic Multiphase Flow in Heterogeneous Media". In: *Water Resources Research* 55.1 (2019), pp. 703–728. DOI: 10.1029/2018WR023528.

[76] Y. Yang and P. Perdikaris. "Conditional deep surrogate models for stochastic, high-dimensional, and multi-fidelity systems". In: *Computational Mechanics* 64.2 (2019), pp. 417–434. DOI: 10.1007/s00466-019-01718-y.

[77] F. Rosenblatt. *Principles of neurodynamics: perceptions and the theory of brain mechanisms*. Spartan, 1962. URL: https://apps.dtic.mil/docs/citations/AD0256582.

[78] D. E. Rumelhart, G. E. Hinton, and P. R. J. Williams. "Learning Representations by Back-propagating Errors". In: *Nature* (1986), pp. 533–536. DOI: 10.1038/323533a0.

[79] J. J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the National Academy of Sciences* 79.8 (1982), pp. 2554–2558. DOI: 10.1073/pnas.79.8.2554.

[80] N. Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[81] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations". In: *arXiv e-print* (2017). URL: https://arxiv.org/pdf/1711.10561.pdf.

[82] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics Informed Deep Learning (Part II): Data-driven Solutions of Nonlinear Partial Differential Equations". In: *arXiv e-print* (2017). URL: https://arxiv.org/pdf/1711.10566.pdf.

[83] M. Raissi. "Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations". In: *Journal of Machine Learning Research* 19.25 (2018), pp. 1–24. URL: http://jmlr.org/papers/v19/18-046.html.

[84] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707. DOI: 10.1016/j.jcp.2018.10.045.

[85] Y. Yang and P. Perdikaris. "Adversarial uncertainty quantification in physics-informed neural networks". In: *Journal of Computational Physics* 394 (2019), pp. 136–152. DOI: 10.1016/j.jcp.2019.05.027.

[86] N. Geneva and N. Zabaras. "Modeling the Dynamics of PDE Systems with Physics-Constrained Deep Auto-Regressive Networks". In: *arXiv e-print* (2019). URL: https://arxiv.org/abs/1906.05747.

[87] C. Grigo and P.-S. Koutsourelakis. "A physics-aware, probabilistic machine learning framework for coarse-graining high-dimensional systems in the Small Data regime". In: *Journal of Computational Physics* 397 (2019). DOI: 10.1016/j.jcp.2019.05.053.

[88] P.-S. Koutsourelakis. "Accurate Uncertainty Quantification Using Inaccurate Computational Models". In: *SIAM Journal on Scientific Computing* 31.5 (2009), pp. 3274–3300. DOI: 10.1137/080733565.

[89] J. Biehler, M. W. Gee, and W. A. Wall. "Towards efficient uncertainty quantification in complex and large-scale biomechanical problems based on a Bayesian multi-fidelity scheme". In: *Biomechanics and Modeling in Mechanobiology* 14.3 (2015), pp. 489–513. DOI: 10.1007/s10237-014-0618-0.

[90] U. Hornung. *Homogenization and Porous Media*. Springer New York, 1997. DOI: 10.1007/978-1-4612-1920-0.

[91] G. Allaire. *Shape Optimization by the Homogenization Method*. Springer New York, 2002. DOI: 10.1007/978-1-4684-9286-6.

[92] G. W. Milton. *The Theory of Composites*. Cambridge University Press, 2002. DOI: 10.1017/CBO9780511613357.

[93] P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer Berlin Heidelberg, 1999. DOI: 10.1007/978-3-662-03939-7.

[94] S. P. Sutera and R. Skalak. "The History of Poiseuille's Law". In: *Annual Review of Fluid Mechanics* 25.1 (1993), pp. 1–20. DOI: 10.1146/annurev.fl.25.010193.000245.

[95] J. Kozeny. "Ueber Kapillare Leitung der Wasser in Boden". In: *Royal Academy of Science, Vienna, Proc. Class I* 136 (1927), pp. 271–306. URL: https://www.zobodat.at/pdf/SBAWW_136_2a_0271-0306.pdf.

[96] P. C. Carman. "Fluid Flow through Granular Beds". In: *Trans. Inst. Chem. Eng.* 15 (1937), pp. 150–166. DOI: 10.1016/s0263-8762(97)80003-2.

[97] P. C. Carman. "Flow of Gases through Porous Media". In: *New York Academic Press* (1956).

[98] G. E. Archie. "Electrical resistivity an aid in core-analysis interpretation". In: *AAPG Bulletin* 31.2 (1947). URL: http://archives.datapages.com/data/bulletns/1944-48/data/pg/0031/0002/0350/0350.htm.

[99] M. Tipping. "Probabilistic Visualisation of High-dimensional Binary Data". In: *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*. MIT Press, 1999, pp. 592–598. URL: https://papers.nips.cc/paper/1561-probabilistic-visualisation-of-high-dimensional-binary-data.

[100] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *arXiv e-print* (2013). URL: http://arxiv.org/abs/1312.6114.

[101] D. J. Rezende, S. Mohamed, and D. Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. 2. 2014, pp. 1278–1286. URL: http://proceedings.mlr.press/v32/rezende14.html.

[102] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, 2002. DOI: 10.1137/1.9780898718027.

[103] C. Grigo and P.-S. Koutsourelakis. "Probabilistic Reduced-Order Modeling for Stochastic Partial Differential Equations". In: *2nd International Conference on Uncertainty Quantification in Computational Sciences and Engineering (UNCECOMP)*. 2017, pp. 111–129. DOI: 10.7712/120217.5356.16731.

[104] C. Grigo and P.-S. Koutsourelakis. "Bayesian Model and Dimension Reduction for Uncertainty Propagation: Applications in Random Media". In: *SIAM /ASA Journal on Uncertainty Quantification* 7.1 (2019), pp. 292–323. DOI: 10.1137/17M1155867.

[105] C. Chauvière, J. Hesthaven, and L. Lurati. "Computational Modeling of Uncertainty in Time-Domain Electromagnetics". In: *SIAM Journal on Scientific Computing* 28.2 (2006), pp. 751–775. DOI: 10.1137/040621673. eprint: 10.1137/040621673. URL: 10.1137/040621673.

[106] G. Bao et al. "An inverse random source problem for the Helmholtz equation". In: *Math. Comput.* 83 (2013), pp. 215–233. DOI: 10.1090/S0025-5718-2013-02730-5.

[107] P. Li and G. Yuan. "Stability on the inverse random source scattering problem for the one-dimensional Helmholtz equation". In: *Journal of Mathematical Analysis and Applications* 450.2 (2017), pp. 872–887. DOI: 10.1016/j.jmaa.2017.01.074.

[108] X. F. Xu and X. Chen. "Stochastic homogenization of random elastic multiphase composites and size quantification of representative volume element". In: *Mechanics of Materials* 41.2 (2009), pp. 174–186. DOI: 10.1016/j.mechmat.2008.09.002.

[109] D. Xiu and G. E. Karniadakis. "Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos". In: *Computer Methods in Applied Mechanics and Engineering* 191.43 (2002), pp. 4927 –4948. ISSN: 0045-7825. DOI: 10.1016/S0045-7825(02)00421-8.

[110] H. P. G. Darcy. *Les Fontaines publiques de la ville de Dijon. Exposition et application des principes à suivre et des formules à employer dans les questions de distribution d'eau, etc*. V. Dalamont, 1856. URL: https://gallica.bnf.fr/ark:/12148/bpt6k624312.image.

[111] J. Fourier. *Théorie analytique de la chaleur, par M. Fourier*. Chez Firmin Didot, père et fils, 1822. DOI: 10.3931/e-rara-19706.

[112] G. S. Ohm. *Die galvanische Kette, mathematisch bearbeitet*. TH Riemann, 1827. URL: http://mdz-nbn-resolving.de/urn:nbn:de:bvb:12-bsb10860421-3.

[113] A. Fick. "Ueber Diffusion". In: *Annalen der Physik* 170.1 (1855), pp. 59–86. DOI: 10.1002/andp.18551700105.

[114] S. Whitaker. ""Flow in porous media I: A theoretical derivation of Darcy's law"". In: *Transport in Porous Media* 1.1 (1986), pp. 3–25. DOI: 10.1007/BF01036523.

[115] E. Sanchez-Palencia. "Non-Homogeneous Media and Vibration Theory". In: *Lecture Notes in Physics* 127 (1980). DOI: 10.1007/3-540-10000-8.

[116] L. Tartar. “Incompressible fluid flow in a porous medium-convergence of the homogenization process”. In: *Appendix of [115]* (1980). DOI: 10.1007/3-540-10000-8.

[117] G. Allaire. “Homogenization of the Stokes flow in a connected porous medium”. In: *Asymptotic Analysis* 2.3 (1989), pp. 203–222. DOI: 10.3233/ASY-1989-2302.

[118] G. Allaire. “Homogenization of the unsteady Stokes equations in porous media”. In: *Progress in partial differential equations: calculus of variations, applications*. Vol. 267. Longman Higher Education, 1992, pp. 109–123. URL: http://www.cmap.polytechnique.fr/~allaire/unsteady-stokes.pdf.

[119] C. Sandström et al. “A two-scale finite element formulation of Stokes flow in porous media”. In: *Computer Methods in Applied Mechanics and Engineering* 261-262 (2013), pp. 96–104. DOI: 10.1016/j.cma.2013.03.025.

[120] V. Marchenko and E. Khruslov. *Homogenization of Partial Differential Equations*. Birkhäuser Boston, 2006. DOI: 10.1007/978-0-8176-4468-0.

[121] L. Tartar. *The General Theory of Homogenization: A Personalized Introduction*. Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-05195-1.

[122] Y. Jiao, F. H. Stillinger, and S. Torquato. “Modeling heterogeneous materials via two-point correlation functions: Basic principles”. In: *Phys. Rev. E* 76 (3 2007). DOI: 10.1103/PhysRevE.76.031110.

[123] J. Feng et al. “Statistical reconstruction of two-phase random media”. In: *Computers & Structures* 137 (2014). Special Issue *UK Association for Computational Mechanics in Engineering*, pp. 78–92. DOI: 10.1016/j.compstruc.2013.03.019.

[124] J. W. Feng et al. “Statistical reconstruction and Karhunen–Loève expansion for multiphase random media”. In: *International Journal for Numerical Methods in Engineering* 105.1 (2016), pp. 3–32. DOI: 10.1002/nme.4957.

[125] M. Shinozuka and G. Deodatis. “Simulation of Multi-Dimensional Gaussian Stochastic Fields by Spectral Representation”. In: *Applied Mechanics Reviews* 49.1 (1996), pp. 29–53. DOI: 10.1115/1.3101883.

[126] S. Bochner. *Lectures on Fourier Integrals*. Princeton University Press, 1959. DOI: 10.1016/b978-044450871-3/50155-8.

[127] M. L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer New York, 1999. DOI: 10.1007/978-1-4612-1494-6.

[128] I. I. Gihman and A. V. Skorokhod. *The theory of stochastic processes I*. Springer, 2004. DOI: 10.1007/978-3-642-61943-4.

[129] J. Fish and T. Belytschko. *A first course in finite elements*. Wiley, 2007. DOI: 10.5860/choice.45-3218.

[130] T. J. R. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[131] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The finite element method: its basis and fundamentals*. Elsevier, 2013. DOI: 10.1016/b978-1-85617-633-0.00020-4.

[132] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods*. Springer, 1991.

[133] M. Crouzeix and P.-A. Raviart. "Conforming and nonconforming finite element methods for solving the stationary Stokes equations I". In: *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique* 7.R3 (1973), pp. 33–75. DOI: 10.1051/m2an/197307r300331.

[134] D. N. Arnold, F. Brezzi, and M. Fortin. "A stable finite element for the Stokes equations". In: *CALCOLO* 21.4 (1984), pp. 337–344. DOI: 10.1007/BF02576171.

[135] C. Taylor and P. Hood. "A numerical solution of the Navier-Stokes equations using the finite element technique". In: *Computers & Fluids* 1.1 (1973), pp. 73–100. DOI: 10.1016/0045-7930(73)90027-3.

[136] B. Delaunay. "Sur la Sphere Vide". In: *Izvestia Akademii Nauk SSSR* 7 (1934), pp. 793–800.

[137] A. Logg, K.-A. Mardal, G. N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012. DOI: 10.1007/978-3-642-23099-8.

[138] H. P. Langtangen and A. Logg. *Solving PDEs in Python: The FEniCS Tutorial I*. Springer Publishing Company, Incorporated, 2017. DOI: 10.1007/978-3-319-52462-7.

[139] T. Mitchell et al. "Machine learning". In: *Annual review of computer science* 4.1 (1990), pp. 417–433. URL: http://profsite.um.ac.ir/~monsefi/machine-learning/pdf/Machine-Learning-Tom-Mitchell.pdf.

[140] F. J. Samaniego. *A Comparison of the Bayesian and Frequentist Approaches to Estimation*. Springer New York, 2010. DOI: 10.1007/978-1-4419-5941-6.

[141] T. Bayes and M. Price. "An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S". In: *Philosophical Transactions of the Royal Society of London* 53 (1763), pp. 370–418. DOI: 10.1098/rstl.1763.0053.

[142] A. N. Tikhonov. "On the stability of inverse problems". In: *Dokl. Akad. Nauk SSSR* 39 (1943), pp. 195–198.

[143] A. E. Hoerl and R. W. Kennard. "Ridge Regression: Biased Estimation for Nonorthogonal Problems". In: *Technometrics* 12.1 (1970), pp. 55–67. DOI: 10.1080/00401706.1970.10488634.

[144] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC Press, 2015. URL: https://web.stanford.edu/~hastie/StatLearnSparsity_files/SLS.pdf.

[145] R. Tibshirani. "Regression Shrinkage and Selection Via the Lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[146] J. H. Friedman, T. Hastie, and R. Tibshirani. "Regularization Paths for Generalized Linear Models via Coordinate Descent". In: *Journal of Statistical Software* 33.i01 (2010). DOI: 10.18637/jss.v033.i01.

[147] M. A. T. Figueiredo. "Adaptive sparseness for supervised learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.9 (2003), pp. 1150–1159. DOI: 10.1109/TPAMI.2003.1227989.

[148] B. Efron et al. "Least angle regression". In: *Ann. Statist.* 32.2 (2004), pp. 407–499. DOI: 10.1214/009053604000000067.

[149] H. Zou, T. Hastie, and R. Tibshirani. "On the "degrees of freedom" of the lasso". In: *Ann. Statist.* 35.5 (2007), pp. 2173–2192. DOI: 10.1214/009053607000000127.

[150] D. J. C. MacKay. "Bayesian Methods for Backpropagation Networks". In: *Models of Neural Networks III: Association, Generalization, and Representation*. Springer New York, 1996, pp. 211–254. DOI: 10.1007/978-1-4612-0723-8_6.

[151] D. J. C. MacKay. "Probable networks and plausible predictions – a review of practical Bayesian methods for supervised neural networks". In: *Network: Computation in Neural Systems* 6.3 (1995), pp. 469–505. DOI: 10.1088/0954-898X_6_3_011.

[152] H. Jeffreys. "An Invariant Form for the Prior Probability in Estimation Problems". In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 186.1007 (1946), pp. 453–461. DOI: 10.1098/rspa.1946.0056.

[153] M. E. Tipping. "Sparse Bayesian learning and the relevance vector machine". In: *Journal of Machine Learning Research* 1 (2001), pp. 211–244. URL: http://www.jmlr.org/papers/v1/tipping01a.html.

[154] R. M. Neal. *Bayesian Learning for Neural Networks*. Springer New York, 1996. DOI: 10.1007/978-1-4612-0745-0.

[155] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006. URL: https://www.springer.com/de/book/9780387310732.

[156] M. E. Tipping. "The Relevance Vector Machine". In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*. 1999, pp. 652–658. URL: http://papers.nips.cc/paper/1719-the-relevance-vector-machine.pdf.

[157] M. E. Tipping. "Relevance vector machine". Pat. 6633857. 2003. URL: http://www.freepatentsonline.com/6633857.html.

[158] J. L. W. V. Jensen. "Sur les fonctions convexes et les inégalités entre les valeurs moyennes". In: *Acta Mathematica* 30 (1906), pp. 175–193. DOI: 10.1007/BF02418571.

[159] A. C. Faul. "Analysis of sparse Bayesian learning". In: *Advances in Neural Information Processing Systems (NIPS)* 14 (2003), pp. 383–389. URL: http://

papers.nips.cc/paper/2121-analysis-of-sparse-bayesian-learning.pdf.

[160] M. Tipping and A. Faul. “Fast Marginal Likelihood Maximisation for Sparse Bayesian Models”. In: *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*. 2003, pp. 3–6. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.4281&rep=rep1&type=pdf.

[161] C. M. Bishop and M. E. Tipping. “Variational Relevance Vector Machines”. In: *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 2000, pp. 46–53. URL: http://miketipping.com/papers/Bishop-VRVM-UAI-00.pdf.

[162] Tipping, M., Bishop, C. “Variational relevance vector machine”. Pat. 6879944. 2005. URL: http://www.freepatentsonline.com/6879944.html.

[163] S. Kullback and R. A. Leibler. “On Information and Sufficiency”. In: *Ann. Math. Statist.* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694.

[164] S. Kullback. *Information theory and statistics*. Dover, 1968. DOI: 10.1090/S0002-9904-1960-10498-3.

[165] N. Reid and D. R. Cox. “On Some Principles of Statistical Inference”. In: *International Statistical Review* 83.2 (2015), pp. 293–308. DOI: 10.1111/insr.12067.

[166] R. E. Kass. “Statistical Inference: The Big Picture”. In: *Statistical Science* 26.1 (2011), pp. 1–9. DOI: 10.1214/10-STS337.

[167] E.-J. Wagenmakers et al. “Bayesian Versus Frequentist Inference”. In: *Bayesian Evaluation of Informative Hypotheses*. Springer New York, 2008, pp. 181–207. DOI: 10.1007/978-0-387-09612-4_9.

[168] G. A. Young and R. L. Smith. *Essentials of Statistical Inference*. Cambridge University Press, 2005. DOI: 10.1017/CBO9780511755392.

[169] L. Held and D. Sabanés Bové. *Applied Statistical Inference: Likelihood and Bayes*. Springer Berlin Heidelberg, 2014. DOI: 10.1007/978-3-642-37887-4.

[170] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. URL: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/38136.pdf.

[171] J. C. Spall. *Introduction to Stochastic Search and Optimization*. 1st ed. John Wiley & Sons, Inc., 2003. DOI: 10.1002/0471722138.

[172] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer New York, 2006. DOI: 10.1007/978-0-387-40065-5.

[173] J. A. Snyman. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Springer US, 2005. DOI: 10.1007/b105200.

[174] A. P. Dempster, N. M. Laird, and D. B. Rubin. “Maximum Likelihood from Incomplete Data Via the EM Algorithm”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22. DOI: 10.1111/j.2517-6161.1977.tb01600.x.

[175] G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions; 2nd ed.* Wiley, 2008. DOI: 10.1002/9780470191613.

[176] P. S. Laplace. "English translation: Memoir on the Probability of the Causes of Events (1774)". In: *Statistical Science* 1.3 (1986), pp. 364–378. URL: http://www.jstor.org/stable/2245476.

[177] L. Isserlis. "On a Formula for the Product-Moment Coefficient of any Order of a Normal Frequency Distribution in any Number of Variables". In: *Biometrika* 12.1/2 (1918), pp. 134–139. DOI: 10.2307/2331932.

[178] K. Binder and D. W. Heermann. *Monte Carlo Simulation in Statistical Physics: An Introduction*. Springer Berlin Heidelberg, 1992. DOI: 10.1007/978-3-662-30273-6.

[179] D. P. Landau and K. Binder. *A Guide to Monte Carlo Simulations in Statistical Physics*. 4th ed. Cambridge University Press, 2014. DOI: 10.1017/CBO9781139696463.

[180] S. Brooks. *Handbook of Markov chain Monte Carlo*. Chapman & Hall/CRC, 2011. DOI: 10.1201/b10905.

[181] D. Gamerman and H. F. Lopes. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference; 2nd ed.* Taylor & Francis, 2006. URL: https://content.taylorfrancis.com/books/download?dac=C2009-0-03659-3&isbn=9780429183348&format=googlePreviewPdf.

[182] C. Andrieu et al. "An Introduction to MCMC for Machine Learning". In: *Machine Learning* 50.1 (2003), pp. 5–43. DOI: 10.1023/A:1020281327116.

[183] N. Metropolis et al. "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6 (1953), pp. 1087–1092. DOI: 10.1063/1.1699114.

[184] W. K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1 (1970), pp. 97–109. DOI: 10.1093/biomet/57.1.97.

[185] V. Ambegaokar and M. Troyer. "Estimating errors reliably in Monte Carlo simulations of the Ehrenfest model". In: *American Journal of Physics* 78.2 (2010), pp. 150–157. DOI: 10.1119/1.3247985.

[186] G. O. Roberts and J. S. Rosenthal. "Optimal scaling for various Metropolis-Hastings algorithms". In: *Statistical Science* 16.4 (2001), pp. 351–367. DOI: 10.1214/ss/1015346320.

[187] S. Duane et al. "Hybrid Monte Carlo". In: *Physics Letters B* 195.2 (1987), pp. 216–222. DOI: 10.1016/0370-2693(87)91197-X.

[188] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer New York, 2004. DOI: 10.1007/978-0-387-76371-2.

[189] M. Girolami and B. Calderhead. "Riemann manifold Langevin and Hamiltonian Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (2011), pp. 123–214. DOI: 10.1111/j.1467-9868.2010.00765.x.

[190] U. Grenander and M. I. Miller. "Representations of Knowledge in Complex Systems". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 56.4 (1994), pp. 549–581. DOI: 10.1111/j.2517-6161.1994.tb02000.x.

[191] G. O. Roberts and R. L. Tweedie. "Exponential convergence of Langevin distributions and their discrete approximations". In: *Bernoulli* 2.4 (1996), pp. 341–363. URL: https://projecteuclid.org:443/euclid.bj/1178291835.

[192] G. O. Roberts and J. S. Rosenthal. "Optimal scaling of discrete approximations to Langevin diffusions". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 60.1 (1998), pp. 255–268. DOI: 10.1111/1467-9868.00123.

[193] D. Rudoy and P. J. Wolfe. "Monte Carlo Methods for Multi-Modal Distributions". In: *2006 Fortieth Asilomar Conference on Signals, Systems and Computers*. 2006, pp. 2019–2023. DOI: 10.1109/ACSSC.2006.355120.

[194] S. Lan, J. Streets, and B. Shahbaba. "Wormhole Hamiltonian Monte Carlo". In: *Proceedings of the ... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence* 2014 (2014), 1953—1959. URL: http://europepmc.org/articles/PMC4386289.

[195] R. H. Swendsen and J.-S. Wang. "Replica Monte Carlo Simulation of Spin-Glasses". In: *Phys. Rev. Lett.* 57 (21 1986), pp. 2607–2609. DOI: 10.1103/PhysRevLett.57.2607.

[196] D. J. Earl and M. W. Deem. "Parallel tempering: Theory, applications, and new perspectives". In: *Phys. Chem. Chem. Phys.* 7 (23 2005), pp. 3910–3916. DOI: 10.1039/B509983H.

[197] G. Altekar et al. "Parallel Metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference". In: *Bioinformatics* 20.3 (2004), pp. 407–415. DOI: 10.1093/bioinformatics/btg427.

[198] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer New York, 2001. DOI: 10.1007/978-1-4757-3437-9.

[199] P. Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer New York, 2004. DOI: /10.1007/978-1-4684-9393-1.

[200] P. Del Moral, A. Doucet, and A. Jasra. "Sequential Monte Carlo samplers". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.3 (2006), pp. 411–436. DOI: 10.1111/j.1467-9868.2006.00553.x.

[201] I. Bilionis and P.-S. Koutsourelakis. "Free energy computations by minimization of Kullback–Leibler divergence: An efficient adaptive biasing potential method for sparse representations". In: *Journal of Computational Physics* 231.9 (2012), pp. 3849 –3870. DOI: 10.1016/j.jcp.2012.01.033.

[202] M. I. Jordan et al. "An Introduction to Variational Methods for Graphical Models". In: *Machine Learning* 37.2 (1999), pp. 183–233. DOI: 10.1023/A:1007665907178.

[203] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002. DOI: 10.5860/choice.41-5949.

[204] M. J. Wainwright and M. I. Jordan. "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends® in Machine Learning* 1.1–2 (2008), pp. 1–305. DOI: 10.1561/2200000001. URL: http://dx.doi.org/10.1561/2200000001.

[205] P. Weiss. "L'hypothèse du champ moléculaire et la propriété ferromagnétique". In: *J. Phys. Theor. Appl.* 6.1 (1907), pp. 661–690. DOI: 10.1051/jphystap:019070060066100. URL: https://hal.archives-ouvertes.fr/jpa-00241247.

[206] P. M. Chaikin and T. C. Lubensky. *Principles of Condensed Matter Physics*. Cambridge University Press, 1995. DOI: 10.1017/CBO9780511813467.

[207] S. Sachdev. "Quantum Phase Transitions". In: *Handbook of Magnetism and Advanced Magnetic Materials*. American Cancer Society, 2007. DOI: 10.1002/9780470022184.hmm108.

[208] C. Peterson and J. Anderson. "A mean field theory learning algorithm for neural networks". In: *Complex Systems* 1 (1987), pp. 995–1019. URL: https://wpmedia.wolfram.com/uploads/sites/13/2018/02/01-5-6.pdf.

[209] L. K. Saul and M. I. Jordan. "Exploiting Tractable Substructures in Intractable Networks". In: *Proceedings of the 8th International Conference on Neural Information Processing Systems*. MIT Press, 1995, pp. 486–492. URL: https://pdfs.semanticscholar.org/e9a5/4374aec5c92296c7b24436f08934643829ae.pdf.

[210] T. S. Jaakkola and M. I. Jordan. "Computing Upper and Lower Bounds on Likelihoods in Intractable Networks". In: *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1996, pp. 340–348. URL: https://arxiv.org/ftp/arxiv/papers/1302/1302.3586.pdf.

[211] J. Paisley, D. M. Blei, and M. I. Jordan. "Variational Bayesian Inference with Stochastic Search". In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. Edinburgh, Scotland: Omnipress, 2012, pp. 1363–1370. URL: https://icml.cc/Conferences/2012/papers/687.pdf.

[212] M. D. Hoffman et al. "Stochastic Variational Inference". In: *Journal of Machine Learning Research* 14 (2013), pp. 1303–1347. URL: http://jmlr.org/papers/v14/hoffman13a.html.

[213] R. Ranganath, S. Gerrish, and D. Blei. "Black Box Variational Inference". In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. Vol. 33. PMLR, 2014, pp. 814–822. URL: http://proceedings.mlr.press/v33/ranganath14.html.

[214] Y. Burda, R. Grosse, and R. Salakhutdinov. "Importance weighted autoencoders". In: *arXiv e-print* (2015). URL: https://arxiv.org/pdf/1509.00519.

[215] H. Robbins and S. Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407. DOI: 10.1214/aoms/1177729586.

[216] S. Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv e-print* (2016). URL: https://arxiv.org/abs/1609.04747.

[217] L. Bottou, F. Curtis, and J. Nocedal. "Optimization Methods for Large-Scale Machine Learning". In: *SIAM Review* 60.2 (2018), pp. 223–311. DOI: 10.1137/16M1080173.

[218] E. Zeidler. "Wichtige Formeln, Graphische Darstellungen und Tabellen". In: *Springer-Taschenbuch der Mathematik: Begründet von I.N. Bronstein und K.A. Semendjaew Weitergeführt von G. Grosche, V. Ziegler und D. Ziegler Herausgegeben von E. Zeidler*. Springer Fachmedien Wiesbaden, 2013, pp. 3–213. DOI: 10.1007/978-3-8348-2359-5_1.

[219] J. Duchi, E. Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." In: *Journal of Machine Learning Research* 12.7 (2011), pp. 2121–2159. URL: http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf.

[220] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv e-print* (2014). URL: http://arxiv.org/abs/1412.6980.

[221] R. Bellman. "Dynamic Programming". In: *Science* 153.3731 (1966), pp. 34–37. DOI: 10.1126/science.153.3731.34.

[222] J. Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (2015), pp. 85 –117. DOI: 10.1016/j.neunet.2014.09.003.

[223] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University, 1975. URL: https://books.google.de/books?id=z81XmgEACAAJ.

[224] Y. LeCun. "Une procedure d'apprentissage pour reseau a seuil asymmetrique (A learning scheme for asymmetric threshold networks)". In: *Proceedings of Cognitiva 85, Paris, France*. 1985, pp. 599–604.

[225] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133. DOI: 10.1007/BF02478259.

[226] Y. Bengio. "Learning Deep Architectures for AI". In: *Foundations and Trends in Machine Learning* 2.1 (2009), pp. 1–127. DOI: 10.1561/2200000006.

[227] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314. DOI: 10.1007/BF02551274.

[228] H. Wu. "Global stability analysis of a general class of discontinuous neural networks with linear growth activation functions". In: *Information Sciences* 179.19 (2009), pp. 3432–3441. DOI: 10.1016/j.ins.2009.06.006.

[229] S. J. Hanson and L. Y. Pratt. "Comparing Biases for Minimal Network Construction with Back-propagation". In: *Proceedings of the 1st International Conference on Neural Information Processing Systems*. MIT Press, 1988, pp. 177–185. URL: https://papers.nips.cc/paper/156-comparing-biases-for-minimal-network-construction-with-back-propagation.

[230] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. "Generalization by Weight-elimination with Application to Forecasting". In: *Proceedings of the 3rd International Conference on Neural Information Processing Systems*. Morgan Kaufmann Publishers Inc., 1990, pp. 875–882. URL: http://papers.nips.cc/paper/323-generalization-by-weight-elimination-with-application-to-forecasting.pdf.

[231] Y. Yao, L. Rosasco, and A. Caponnetto. "On early stopping in gradient descent learning". In: *Constr. Approx* (2007), pp. 289–315. DOI: 10.1007/s00365-006-0663-2.

[232] G. E. Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv e-print* (2012). URL: http://arxiv.org/abs/1207.0580.

[233] J. Hensman, N. Fusi, and N. D. Lawrence. "Gaussian Processes for Big Data". In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. Bellevue, WA: AUAI Press, 2013, pp. 282–290. URL: https://arxiv.org/ftp/arxiv/papers/1309/1309.6835.pdf.

[234] J. de Baar, R. Dwight, and H. Bijl. "Speeding up Kriging through fast estimation of the hyperparameters in the frequency-domain". In: *Computers & Geosciences* 54 (2013), pp. 99–106. DOI: 10.1016/j.cageo.2013.01.016.

[235] E. Snelson and Z. Ghahramani. "Sparse Gaussian Processes Using Pseudo-inputs". In: *Proceedings of the 18th International Conference on Neural Information Processing Systems*. 2005, pp. 1257–1264. URL: http://papers.nips.cc/paper/2857-sparse-gaussian-processes-using-pseudo-inputs.pdf.

[236] T. Muehlenstaedt et al. "Data-driven Kriging models based on FANOVA-decomposition". In: *Statistics and Computing* 22.3 (2012), pp. 723–738. DOI: 10.1007/s11222-011-9259-7.

[237] N. D. Lawrence. "Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data". In: *Proceedings of the 16th International Conference on Neural Information Processing Systems*. MIT Press, 2003, pp. 329–336. URL: https://papers.nips.cc/paper/2540-gaussian-process-latent-variable-models-for-visualisation-of-high-dimensional-data.pdf.

[238] C. Soize and R. Ghanem. "Physical Systems with Random Uncertainties: Chaos Representations with Arbitrary Probability Measure". In: *SIAM Journal on Scientific Computing* 26.2 (2004), pp. 395–410. DOI: 10.1137/S1064827503424505.

[239] D. Xiu and J. Hesthaven. "High-Order Collocation Methods for Differential Equations with Random Inputs". In: *SIAM Journal on Scientific Computing* 27.3 (2005), pp. 1118–1139. DOI: 10.1137/040615201.

[240] S. A. Smolyak. "Quadrature and interpolation formulas for tensor products of certain classes of functions". In: 1963. URL: http://www.mathnet.ru/links/1e51c83614c1a32e60ef42245b2eb5d0/dan27586.pdf.

[241] M. Berveiller, B. Sudret, and M. Lemaire. "Stochastic finite element: a non intrusive approach by regression". In: *European Journal of Computational Mechanics* 15.1-3 (2006), pp. 81–92. DOI: 10.3166/remn.15.81-92.

[242] A. Doostan and H. Owhadi. "A non-adapted sparse approximation of PDEs with stochastic inputs". In: *Journal of Computational Physics* 230.8 (2011), pp. 3015–3034. DOI: 10.1016/j.jcp.2011.01.002.

[243] L. Yan, L. Guo, and D. Xiu. "Stochastic collocation algorithms using L_1 minimization". In: *International Journal for Uncertainty Quantification* 2.3 (2012), pp. 279–293. DOI: 10.1615/Int.J.UncertaintyQuantification.2012003925.

[244] D. Xiu and G. E. Karniadakis. "Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos". In: *Computer Methods in Applied Mechanics and Engineering* 191.43 (2002), pp. 4927–4948. DOI: 10.1016/S0045-7825(02)00421-8.

[245] D. Xiu and S. J. Sherwin. "Parametric uncertainty analysis of pulse wave propagation in a model of a human arterial network". In: *Journal of Computational Physics* 226.2 (2007), pp. 1385–1407. DOI: 10.1016/j.jcp.2007.05.020.

[246] C. Eckart and G. Young. "The approximation of one matrix by another of lower rank". In: *Psychometrika* 1.3 (1936), pp. 211–218. DOI: 10.1007/BF02288367.

[247] C. Prud'Homme et al. "Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods". In: *Journal of Fluids Engineering* 124.1 (2001), pp. 70–80. DOI: 10.1115/1.1448332.

[248] X. Yang, G. Tartakovsky, and A. Tartakovsky. "Physics-Informed Kriging: A Physics-Informed Gaussian Process Regression Method for Data-Model Convergence". In: *arXiv e-print* (2018). URL: https://arxiv.org/pdf/1809.03461.pdf.

[249] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. English. 1 edition. The MIT Press, 2009. URL: https://mitpress.mit.edu/books/probabilistic-graphical-models.

[250] L. Felsberger and P.-S. Koutsourelakis. "Physics-Constrained, Data-Driven Discovery of Coarse-Grained Dynamics". In: *Communications in Computational Physics* 25.5 (2019), pp. 1259–1301. DOI: 10.4208/cicp.OA-2018-0174.

[251] M.Schöberl, N.Zabaras, and P.-S.Koutsourelakis. "*Predictive* coarse-graining". In: *Journal of Computational Physics* 333 (2017), pp. 49–77. DOI: 10.1016/j.jcp.2016.10.073.

[252] M. Schöberl, N. Zabaras, and P.-S. Koutsourelakis. "Predictive collective variable discovery with deep Bayesian models". In: *The Journal of Chemical Physics* 150.2 (2019). DOI: 10.1063/1.5058063.

[253] N. Tishby, F. C. Pereira, and W. Bialek. "The Information Bottleneck Method". In: *Proc. of the 37th Annual Allerton Conference on Communication, Control and Computing*. 1999, pp. 368–377. URL: https://arxiv.org/abs/physics/0004057.

[254] Y. LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[255] S. Lowell et al. *Characterization of Porous Solids and Powders: Surface Area, Pore Size and Density*. Vol. 1. Springer, 2006. DOI: 10.1007/978-1-4020-2303-3.

[256] B. Lu and S. Torquato. "Lineal-path function for random heterogeneous materials". In: *Phys. Rev. A* 45 (1992), pp. 922–929. DOI: 10.1103/PhysRevA.45.922.

[257] S. Torquato and G. Stell. "Microstructure of two-phase random media. I. The n-point probability functions". In: *The Journal of Chemical Physics* 77.4 (1982), pp. 2071–2077. DOI: 10.1063/1.444011.

[258] J. C. Michel, H. Moulinec, and P. Suquet. "Effective properties of composite materials with periodic microstructure: a computational approach". In: *Computer Methods in Applied Mechanics and Engineering* 172.1 (1999), pp. 109–143. DOI: 10.1016/S0045-7825(98)00227-8.

[259] J. C. Maxwell. *A treatise on electricity and magnetism*. Clarendon Press, 1873. URL: http://www.aproged.pt/biblioteca/MaxwellI.pdf.

[260] D. A. G. Bruggeman. "Berechnung verschiedener physikalischer Konstanten von heterogenen Substanzen.I. Dielektrizitätskonstanten und Leitfähigkeiten der Mischkörper aus isotropen Substanzen". In: *Annalen der Physik* 416.7 (1935), pp. 636–664. DOI: 10.1002/andp.19354160705.

[261] Z. Hashin and S. Shtrikman. "A variational approach to the theory of the elastic behaviour of multiphase materials". In: *Journal of the Mechanics and Physics of Solids* 11.2 (1963), pp. 127–140. DOI: 10.1016/0022-5096(63)90060-7.

[262] S. Torquato and B. Lu. "Chord-length distribution function for two-phase random media". In: *Phys. Rev. E* 47 (1993), pp. 2950–2953. DOI: 10.1103/PhysRevE.47.2950.

[263] G. E. Archie. "The electrical resistivity log as an aid in determining some reservoir characteristics". In: *Trans. Am. Inst. Mech. Eng.* 146 (1942), pp. 54–62. DOI: 10.2118/942054-G.

[264] R. Landauer. “The Electrical Resistance of Binary Metallic Mixtures”. In: *Journal of Applied Physics* 23.7 (1952), pp. 779–784. DOI: 10.1063/1.1702301.

[265] R. Landauer. “Electrical conductivity in inhomogeneous media”. In: *AIP Conference Proceedings* 40.1 (1978), pp. 2–45. DOI: 10.1063/1.31150.

[266] P. Debye, H. R. Anderson, and H. Brumberger. “Scattering by an Inhomogeneous Solid. II. The Correlation Function and Its Application”. In: *Journal of Applied Physics* 28.6 (1957), pp. 679–683. DOI: 10.1063/1.1722830.

[267] N. D. Ngo and K. K. Tamma. “Microscale permeability predictions of porous fibrous media”. In: *International Journal of Heat and Mass Transfer* 44.16 (2001), pp. 3135–3145. DOI: 10.1016/S0017-9310(00)00335-5.

[268] I. The MathWorks. “Image processing toolbox”. In: Matlab 2019b ().

[269] R. Kohavi and G. H. John. “Wrappers for feature subset selection”. In: *Artificial Intelligence* 97.1 (1997). Relevance, pp. 273–324. DOI: 10.1016/S0004-3702(97)00043-X.

[270] P. Narendra and K. Fukunaga. “A Branch and Bound Algorithm for Feature Subset Selection”. In: *IEEE Transactions on Computers* 26.09 (1977), pp. 917–922. DOI: 10.1109/TC.1977.1674939.

[271] S. Della Pietra, V. Della Pietra, and J. Lafferty. “Inducing features of random fields”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.4 (1997), pp. 380–393. DOI: 10.1109/34.588021.

[272] C. Hans. “Bayesian lasso regression”. In: *Biometrika* 96.4 (2009), pp. 835–845. DOI: 10.1093/biomet/asp047.

[273] S. Arlot and A. Celisse. “A survey of cross-validation procedures for model selection”. In: *Statistics Surveys* 4 (2010), pp. 40–79. DOI: 10.1214/09-SS054.

[274] H. Akaike. “Information Theory and an Extension of the Maximum Likelihood Principle”. In: *Selected Papers of H. Akaike*. Springer New York, 1998, pp. 199–213. DOI: 10.1007/978-1-4612-1694-0_15.

[275] H. Akaike. “A New Look at the Statistical Model Identification”. In: *Selected Papers of H. Akaike*. Springer New York, 1998, pp. 215–222. DOI: 10.1007/978-1-4612-1694-0_16.

[276] C. M. Stein. “Estimation of the Mean of a Multivariate Normal Distribution”. In: *The Annals of Statistics* 9.6 (1981), pp. 1135–1151. DOI: 10.1214/aos/1176345632.

[277] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. “Variational Inference: A Review for Statisticians”. In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877. DOI: 10.1080/01621459.2017.1285773.

[278] M. J. Beal and Z. Ghahramani. “The Variational Bayesian EM Algorithm for Incomplete Data: with Application to Scoring Graphical Model Structures”. In: *Bayesian Statistics* 7 (2003). URL: http://www.gatsby.ucl.ac.uk/~beal/papers/v7score.pdf.

[279] Z. C. Holcomb. *Fundamentals of descriptive statistics*. Taylor & Francis, 1997. DOI: 10.4324/9781315266510.

[280] D. Zhang. "A Coefficient of Determination for Generalized Linear Models". In: *The American Statistician* 71.4 (2017), pp. 310–316. DOI: 10.1080/00031305.2016.1256839.

[281] B. N. Datta. *Numerical Linear Algebra and Applications, Second Edition*. 2nd. Society for Industrial and Applied Mathematics, 2010. URL: https://archive.siam.org/books/ot116/.

[282] E. Brochu, V. M. Cora, and N. de Freitas. *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*. Tech. rep. 2010. URL: https://arxiv.org/abs/1012.2599.

[283] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. "Predictive Entropy Search for Efficient Global Optimization of Black-box Functions". In: *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, pp. 918–926. URL: http://papers.nips.cc/paper/5324-predictive-entropy-search-for-efficient-global-optimization-of-black-box-functions.pdf.

[284] R. K. Srivastava, K. Greff, and J. Schmidhuber. "Highway networks". In: *arXiv e-print* (2015). URL: https://arxiv.org/abs/1505.00387.

[285] J. G. Zilly et al. "Recurrent Highway Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 4189–4198. URL: http://proceedings.mlr.press/v70/zilly17a.html.

[286] P. Forchheimer. "Wasserbewegung durch Boden". In: *Zeitschrift des Vereins deutscher Ingenieure* 45 (1901), pp. 1782–1788.

[287] D. D. Joseph, D. A. Nield, and G. Papanicolaou. "Nonlinear equation governing flow in a saturated porous medium". In: *Water Resources Research* 18.4 (1982), pp. 1049–1052. DOI: 10.1029/WR018i004p01049.

[288] H. C. Brinkman. "A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles". In: *Flow, Turbulence and Combustion* 1.1 (1949). DOI: 10.1007/BF02120313.

[289] O. Chapelle, B. Schölkopf, and A. Zien. "Semi-Supervised Learning". In: *IEEE Transactions on Neural Networks* 20.3 (2009). DOI: 10.1109/TNN.2009.2015974.

[290] S. Yu et al. "Supervised Probabilistic Principal Component Analysis". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2006, pp. 464–473. DOI: 10.1145/1150402.1150454.

[291] X. Gao et al. "Supervised Gaussian Process Latent Variable Model for Dimensionality Reduction". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 41.2 (2011), pp. 425–434. DOI: 10.1109/TSMCB.2010.2057422.

[292] M. E. Tipping and C. M. Bishop. “Probabilistic Principal Component Analysis”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999), pp. 611–622. DOI: 10.1111/1467-9868.00196.

[293] S. Roweis and Z. Ghahramani. “A Unifying Review of Linear Gaussian Models”. In: *Neural Computation* 11.2 (1999), pp. 305–345. DOI: 10.1162/089976699300016674.