Inaugural Dissertation

submitted to the
Combined Faculty for the Natural Sciences and Mathematics
of Heidelberg University, Germany

for the degree of
Doctor of Natural Sciences

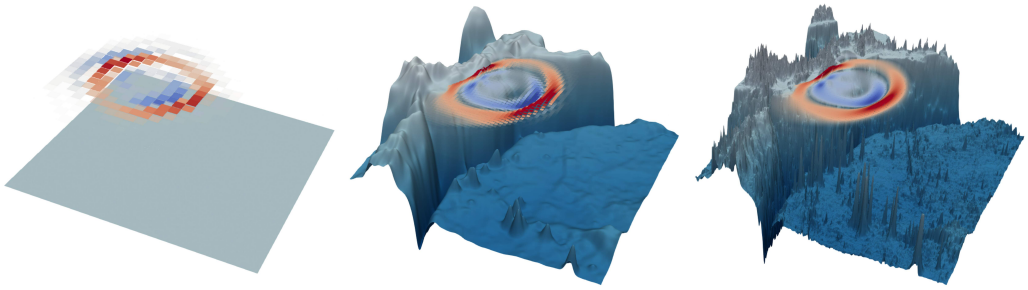
Multiscale Methods for High Performance Uncertainty Quantification

Linus Seelinger

Author: Linus Seelinger, MSc. Scientific Computing
Born in: Speyer
Advisors: Prof. Dr. Peter Bastian and Prof. Dr. Robert Scheichl
Oral examination:

Multiscale Methods for High Performance Uncertainty Quantification

Linus Seelinger



Advisors: Prof. Dr. Peter Bastian and Prof. Dr. Robert Scheichl

September 2021

Linus Seelinger
Multiscale Methods for High Performance Uncertainty Quantification
Doctoral Thesis in Applied Mathematics, September 2021
Advisors: Prof. Dr. Peter Bastian and Prof. Dr. Robert Scheichl

Heidelberg University
Combined Faculty for the Natural Sciences and Mathematics

Abstract

Mathematical models of complex real-world phenomena result in computational challenges, often necessitating the use of modern High Performance Computing (HPC) systems and therefore parallelization. When solving Uncertainty Quantification (UQ) problems on such models, these challenges only increase: Uncertainties in input data or (in case of inverse problems) in measurements essentially contribute to the overall dimensionality of the problem at hand.

This dissertation aims to close the gap between advanced models and advanced UQ methods by three approaches: A parallelization scheme for an efficient hierarchical inverse UQ method is devised, allowing to leverage the full potential of HPC systems; efficient model hierarchies based on Localized Model Order Reduction (LMOR) are investigated, allowing automatic generation of coarse models; and the resulting tools are made available to the wider community as part of the modular and open source MIT Uncertainty Quantification Library (MUQ).

Zusammenfassung

Mathematische Modelle komplexer Phänomene der realen Welt führen zu herausfordernden numerischen Problemen, die oft nur mithilfe von Hochleistungsrechnern unter Parallelisierung gelöst werden können. Uncertainty Quantification (UQ) auf solchen Modellen erhöht diese Schwierigkeiten weiter: Unsicherheiten in Parametern oder (im Fall von inversen Problemen) in Messdaten erhöhen letztlich die Dimensionalität des gesamten Problems.

Diese Dissertation zielt darauf ab, mit den folgenden drei Ansätzen zur Schließung der Lücke zwischen fortschrittlichen UQ-Methoden und Modellen beizutragen: Eine massiv parallelisierte Version eines effizienten hierarchischen UQ-Verfahrens für inverse Probleme wird vorgestellt, was die Nutzung von Hochleistungsrechnern möglich macht; effiziente Modellhierarchien basierend auf Localized Model Order Reduction (LMOR) werden untersucht, die das automatische Generieren vergröberter Modelle erlauben; und die dabei entwickelten Software-Werkzeuge werden für die Gemeinschaft als Teil der modularen und quelloffenen MIT Uncertainty Quantification Library (MUQ) bereitgestellt.

Acknowledgement

First of all, I would like to thank my family: My parents for their loving support, in particular my late father for supporting my interest in math and computing early on, and my wonderful wife for making my life better in so many different ways (and keeping me from eating toast all the time).

I would like to thank Peter Bastian, Ole Klein and Robert Scheichl for their supervision and mentoring. Their guidance was extremely helpful throughout my work.

Further, I would like to thank Peter's and Robert's research groups for a great working environment and many fruitful discussions. In particular, I would like to thank Dominic Kempf, René Heß and Steffen Müthing for frequent productive exchange regarding the DUNE numerical software project.

Anne Reinarz has my gratitude for years of both enjoyable and fruitful collaboration on the CerTest project relating to GenEO as well as uncertainty quantification on ExaHyPE models. Likewise, since he joined, Jean Bénézech has become a highly valued collaborator on CerTest, and I am looking forward to further work on applying model order reduction in that setting.

For their collaboration on model order reduction, I would like to thank Christian Engwer, Andreas Buhr and Chupeng Ma. It has been a great time working on our joint project, and discovering direct links between model order reduction and preconditioning lead to exciting results.

On the other side of my project (and of the pond), I would like to thank Andrew Davis and Matthew Parno for fun and extremely useful collaboration on the MUQ uncertainty quantification library. I am looking forward to future collaborations we are currently planning.

I am also grateful to the administration staff at Heidelberg University for always making things work, in particular Felicitas Hirsch, Herta Fitzer and Joachim Simon.

Finally, I would like to thank the bwHPC BwForCluster MLS&WISO and SuperMUC-NG computing services for providing the computational resources needed to produce my results. Their acknowledgement strings are therefore included at the end.

Contents

1 Introduction

> *I think it's much more interesting to live not knowing than to have answers which might be wrong. I have approximate answers and possible beliefs and different degrees of uncertainty [...]*
>
> — **Richard P. Feynman**

1.1 Motivation and Problem Statement

The above quote by Richard Feynman, an outstanding communicator of the scientific method, illustrates a very fundamental principle: No observation, no model and no prediction about reality will ever be fully accurate. Every measurement is limited in accuracy by the device used to obtain it, and subtle aspects of reality may have been missed in building seemingly good models.

This has been accounted for since the early days of rigorous science: Measurement results are equipped with indications of their degree of reliability, and models by their very name do not claim to be an absolute match of reality.

But what about model predictions? Clearly, they too are strongly affected by those uncertainties, as weather forecasts remind us frequently. Historically, that aspect was somewhat neglected, and understandably so: The computational resources needed to solve complex models for realistic scenarios, let alone to treat stochastic behavior appropriately, only became available over the last few decades.

Consequently, the field of Uncertainty Quantification (UQ), aiming for the computationally efficient treatment of uncertainties in complex models, is fairly new. Drawing from fields concerned with the efficient solution of complex models as well as stochastics, it can reasonably be seen as interdisciplinary. However, a certain gap between the disciplines can be observed in current research: Highly complex simulations are often outfitted with simpler statistical methods, while more advanced statistical methods are often only applied to rather simple problems.

Considering that advanced simulation tools are of ever increasing importance in science and engineering, ranging from the exploration of theoretical models all the way to safety-critical engineering applications, it is clear that closing this gap is essential.

The aim of this thesis is to improve that situation by three approaches: Advancing the efficiency of UQ methods geared towards large-scale models by employing model hierarchies, improving techniques for constructing such model hierarchies and providing an open, modular and model-agnostic technological basis to support the application and further development of those methods in future research.

1.2 Contributions

At its core, this dissertation contains the following contributions to current research:

- Parallelization strategies for Multilevel Markov Chain Monte Carlo (MLMCMC) and Multiindex Markov Chain Monte Carlo (MIMCMC) as well as a modular software framework supporting Multilevel Monte Carlo (MLMC), MLMCMC and MIMCMC in a sequential and, most importantly, in a highly scalable parallel setting. We demonstrate parallel efficiency in practical applications. The software framework is made publicly available as part of the MIT Uncertainty Quantification Library (MUQ) and intends to make those methods available to a wider community.
- An extension of an MLMCMC method for the efficient solution of inverse UQ problems to a more general multiindex case. While the general feasibility of such a method is straightforward and has been proven before, we propose a strategy for using samples from coarser models to accelerate the method on finer models in analogy to an established multilevel method. This allows for gains in efficiency when solving inverse problems in case multiple independent coarsening strategies exist for the model.
- Coupling of these methods to realistic large-scale models demonstrating their fitness for practically relevant High Performance Computing (HPC) applications.
- The discovery of a theoretical link between Model Order Reduction (MOR) methods and the Generalized Eigenproblems in the Overlaps (GenEO) coarse space originating from robust preconditioning of Partial Differential Equation

(PDE) solvers (joint work with Chupeng Ma and Andreas Buhr). This gives a theoretical justification for employing a sightly modified version of GenEO as a model order reduction method. Further, we can exchange methods between the two fields, and demonstrate practical gains in GenEO preconditioner robustness when applying a randomized eigensolver originally designed for MOR methods.

- Development of coarsening strategies suitable for hierarchical UQ methods. As part of the Tsunami application (joint work with Anne Reinarz), we investigate changing the mathematical model itself for PDE solver performance. Working towards a composite materials application (joint work with Jean Bénézech), we propose an online/offline approach reusing large parts of an MOR model for subsequent samples. While each of these approaches is specific to a certain type of model, they substantially increase efficiency in solving inverse problems, and may be transferred to other applications as well.
- Reimplementation of GenEO components in the Distributed and Unified Numerics Environment (DUNE) to support unstructured grids (joint work with Peter Bastian). This is an extension to previous work published by the author in [SRS20]. Due to restrictions of DUNE grid implementations, communication on overlapping grids as needed for GenEO is only directly available on structured grids. We overcome this by inferring the overlaps from discretization matrices constructed on non-overlapping subdomains. As a result, GenEO in DUNE can be applied to advanced engineering problems.
- Various further contributions to the MUQ and DUNE projects.

1.3 Thesis Structure

Chapter 2: Basics of Uncertainty Quantification

We begin with a quick review of basic stochastic concepts underlying the following chapters, and introduce the main types of UQ problems to be treated in a general way. As the foundations of the more advanced methods presented later, we introduce the most basic Monte Carlo (MC) and Markov Chain Monte Carlo (MCMC) methods for solving these problems.

Chapter 3: Advanced Uncertainty Quantification Methods

Here we present advanced multilevel and multiindex versions of the above algorithms. By exploiting model hierarchies, these algorithms allow the treatment of far more challenging UQ problems than their respective basic forms. While analogous to multigrid methods in PDE solvers, they are very general regarding the choice of model hierarchy.

Chapter 4: Efficient Models and Model Hierarchies for Multiscale Problems

Creating suitable model hierarchies for multilevel and multiindex UQ methods allows for many gains in computational efficiency. We give an overview of different approaches and present a specific choice. Making use of a GenEO coarse space originating from preconditioning by interpreting it as a Localized Model Order Reduction (LMOR) method, we can demonstrate significant gains in efficiency using an online / offline approach.

Chapter 5: Modular and Parallel HPC Implementation

In order to effectively apply the multilevel and multiindex methods from Chapter 3 to large scale UQ problems on computationally challenging models, we need to employ modern HPC systems. As a result, massive parallelization is required. In the setting of inverse problems, this is far from trivial since MLMCMC and MIMCMC type methods introduce numerous data dependencies. We present a highly scalable and model agnostic implementation by highlighting its most relevant design decisions and modular architecture.

Chapter 6: Applications

Here we demonstrate real-world effectiveness of our methods and software packages on large-scale HPC applications. We investigate the algorithms' behaviors as well as parallel scalability.

Chapter 7: Conclusion

In the last chapter, we finally give a comprehensive conclusion to this thesis and its results as well as an outlook on possible future research opportunities.

1.4 Reading Guide

The following section intends to guide readers to the aspects of this thesis that should be most relevant to them.

Modeling experts might be most interested in the applications in chapter 6 that show how the UQ methods investigated here can be used to answer questions in realistic scenarios. A particularly relevant point is the simple interface between forward models and UQ methods. This interface can be carried over into software implementations and, when implemented in a modular framework as presented in chapter 5, allows straightforward application of numerous existing UQ methods to the same model. Further, parallel scalability on the UQ side as presented here can be exploited to solve challenging UQ problems without investing too much in the scalability of the forward model itself. Chapter 4 may serve as inspiration when it comes to finding and exploiting hierarchies in forward models to accelerate hierarchical UQ methods.

UQ method developers on beginner level should start with the UQ basics in chapter 2. For more advanced readers, the MLMCMC and MIMCMC methods covered in chapter 3 should be the most interesting regarding methods and algorithms. The applications in chapter 6 demonstrate their effectiveness in practice. Further, the implementation chapter in chapter 5 may serve as motivation to conduct implementations of new methods in existing modular frameworks in order to build on top of existing work where possible, and have an interface that can easily couple to existing forward models.

HPC experts and computer scientists might be most interested in the parallel architecture devised in chapter 5 to facilitate large-scale UQ solvers, as well as the scalability results demonstrating its effectiveness in chapter 6. Further, the online/offline approach to localized model order reduction in chapter 4 highlights opportunities in solving approximations of numerous related large-scale PDE problems at very low computational cost.

To **MOR experts**, we recommend the theoretical link between robust and scalable preconditioning of PDE solvers and LMOR presented in chapter 4. There we show that methods originating from both areas are essentially equivalent and both theoretical and practical tools may be exchanged between the fields. Further, the hierarchical UQ methods introduced in chapter 3 and demonstrated in practice in chapter 6 may provide inspiration in how to use MOR methods in the construction of model hierarchies accelerating UQ solvers.

Finally, **numerical PDE experts** may also be interested in the link between preconditioning and MOR in chapter 4 as well as methods from MOR that can be transferred. More generally, that chapter details how PDEs offer numerous opportunities for constructing model hierarchies for fast UQ solvers. Further, the abstract interface between forward models and UQ as shown in the implementations in chapter 5 can easily be exploited to apply existing UQ methods and codes to a specific PDE model.

2 Basics of Uncertainty Quantification

Physical processes are often affected by a wide variety of uncertainties. This fully and immediately transfers to mathematical models aiming to accurately replicate them. Beyond that, even models themselves may add further uncertainties.

The importance of treating uncertainty in models cannot be overstated, since a seemingly good but actually unreliable result can easily have fatal consequences in safety-critical applications. In fact, one might argue that no result is actually valuable until its reliability can be quantified.

Some frequent types of such uncertainties are parameters only known up to a certain confidence, measurement devices introducing errors, model predictions that can only be computed approximately, or chaotic physical processes. Since they may have significant effects on results, these uncertainties need to be accounted for when constructing and employing models.

Expressing and analyzing uncertainty in mathematical models necessitates a consistent theoretical framework. Whether the uncertainty in question is of aleatoric or epistemic nature, the well-established probability theory is a suitable choice.

In the aleatoric case, where parts of the model itself are considered to be truly random, the model's inherent randomness directly coincides with the notion of random events in probability theory. In case of epistemic uncertainty, where some properties are assumed to have a true value but cannot be determined exactly, probability distributions can be employed to represent our confidence in parameters, model predictions or measurements.

We begin with a brief review of physical models based on a Partial Differential Equation (PDE) in section 2.1. Section 2.2 presents the probabilistic basics needed to express uncertainty in models, which we then use to introduce abstract Uncertainty Quantification (UQ) problems in section 2.3. We proceed to introduce basic UQ algorithms in section 2.4 to numerically solve such problems, and demonstrate them on a simple example. These algorithms will be extended in chapter 3 to significantly improve computational efficiency.

2.1 Physical Models

Physical processes are often suitably described in terms of a PDE. Later we will exploit certain properties of such models in order to construct efficient UQ methods. Therefore we give a brief review of how PDE models can be derived from simple assumptions on the underlying physics, present an abstract Finite element (FE) method to numerically solve them and introduce all notation we require in the following. This section is based on [Bas12; BS07].

As a specific example, we consider subsurface fluid flow, a simple yet in many regards representative case. The state of a fluid can accurately be described in terms of the position and velocity of each individual atom or molecule. In order to compute the fluid's evolution over time, we need to take into account the interactions between each of these particles as well as possible outside forces. When looking at large-scale systems, like predicting the groundwater supply in an entire region across decades or centuries, this approach is entirely intractable from a computational point of view: Simply storing the state of each particle in that large system is far out of reach for any computer available, let alone computing the enormous number of interactions. Even worse, we would have to resolve extremely short time scales to capture particle interactions.

Fortunately however, for many practical applications, resolving each individual particle is not of interest at all. Realistic questions might be: Will a certain well provide enough water for a town? As an answer to that question a well output expressed in cubic meters per second will clearly be more appropriate than an enormous set of particle states. As it turns out, for the model itself it is also often enough to consider quantities on macroscopic scale, for example densities or velocities, instead of tracking particle states on the microscopic scale. We assume that these macroscopic quantities can be accurately represented in terms of (often) continuous and smooth functions.

Derivation of a simple PDE

In case of fluid flow, we may consider the fluid's density $\rho : \Omega \times \Sigma \to \mathbb{R}$ on a spatial domain $\Omega \subset \mathbb{R}^3$ and time interval $\Sigma \subset \mathbb{R}$. The fluid's mass $M_\omega(t)$ at time t in a subdomain $\omega \subset \Omega$ is then simply given by the integral

$$M_\omega(t) = \int_\omega \rho(x,t)dx.$$

This is now a model of the macroscopic state of the fluid in space and time. In order to make this an actually useful model, we still need to model the evolution of the system's state over time. As it turns out, simply assuming conservation of mass is already sufficient to imply the system's temporal behavior. Conservation of mass means, in this case, that M_ω may only change if mass is injected or removed within ω or if mass flows across the boundary $\partial\omega$. Introducing the velocity $v : \Omega \times \Sigma \to \mathbb{R}^3$, we can express that with the equation

$$M_\omega(t+\delta t) - M_\omega(t) = \int_t^{t+\delta t} \left(\int_\omega f(x,r)dx - \int_{\partial\omega} \rho(x,r)v(x,r)\cdot n(x)ds \right) dr \quad (2.1)$$

where δt is an arbitrary time step and n is the unit outer normal to ∂w. Further, f is a source function representing mass added or removed, for example through wells.

We may now use $\int_t^{t+\delta t} g(r)dr = \delta t g(t) + \mathcal{O}(\delta t^2)$ on the left hand side of eq. (2.1) assuming sufficient smoothness of M_ω and further use Gauß' theorem $\int_\omega \nabla \cdot u dx = \int_{\delta\omega} u \cdot n ds$ to convert the boundary integral into a domain integral. Passing to the limit, we arrive at

$$\partial_t \int_\omega \rho(x,t)dx + \int_\omega \nabla\cdot(\rho(x,t)v(x,t))dx = \int_\omega f(x,t)dx.$$

Since the above equation holds for any subdomain ω, we can follow the argument in [Smi86, Section 74] to conclude the differential form of the mass conservation law

$$\partial_t \rho(x,t) + \nabla\cdot(\rho(x,r)v(x,r)) = f(x,t), \qquad x\in\Omega, t\in\Sigma. \quad (2.2)$$

This equation further simplifies due to $\partial_t \rho = 0$ in case of incompressible fluids and we arrive at

$$\nabla\cdot v(x,r) = f(x,t), \qquad x\in\Omega.$$

From mass conservation to subsurface flow

In order to model fluid flow in soil, or more general in porous media, it is often enough to apply the same reasoning as above. Note that by defining the system state only by density, we ignore the fluid's momentum that critically contributes to vorticity. However, since flow in porous media tends to occur at very low velocities, we allow ourselves to ignore momentum in favor of achieving a more simple model.

Modeling porosity, the fraction of the porous material's volume available to fluid flow, we augment eq. (2.2) by a porosity function $\Phi : \Omega \to (0, 1)$.

$$\partial_t(\Phi(x)\rho(x,t)) + \nabla \cdot (\rho(x,r)v(x,r)) = f(x,t), \qquad x \in \Omega, t \in \Sigma. \tag{2.3}$$

Darcy's law relates fluid velocity to pressure gradients via

$$v = -\frac{K}{\mu}(\nabla p - \rho g),$$

where K is the permeability tensor modeling pore structure of the porous medium, μ is the dynamic viscosity of the fluid and g is the gravity vector. Inserting this into eq. (2.3), we arrive at

$$\partial_t(\Phi\rho) - \nabla \cdot (\rho\frac{K}{\mu}(\nabla p - \rho g)) = f \qquad in \quad \Omega \times \Sigma.$$

Further assuming a constant density $\rho(t) = \rho$, i.e. an incompressible fluid, we obtain

$$-\nabla \cdot (\rho\frac{K}{\mu}(\nabla p - \rho g)) = f/\rho \qquad in \quad \Omega \times \Sigma, \tag{2.4}$$

where the only unknown is pressure.

In section 6.1 we will use this model in a simplified form, where assuming zero gravity we arrive at Poisson's equation

$$-\nabla \cdot (\kappa \nabla u) = f \qquad in \quad \Omega \tag{2.5}$$

where we solve for u given parameters κ and f. This serves as a frequently used model problem in numerous fields of applied mathematics, since it exposes properties

representative of a large class of PDE models despite its simplicity. The derivation of the other models in chapter 4 and chapter 6 is outside the scope of this work, but can be conducted in a similar way.

Types of PDEs

Linear PDEs of second order (i.e. ones consisting of at most second derivatives) such as the one above may, assuming continuous differentiability in the coefficients, be expressed in a general form:

$$Lu = -\sum_{i,j=1}^{n} a_{ij}(x)\partial_{x_j}\partial_{x_i}u + \sum_{i=1}^{n} b_i(x)\partial_{x_i}u + c(x)u = f \qquad in \quad \Omega, \tag{2.6}$$

where L is called the differential operator. It can be classified in the following way:

Definition 1 (Classification of second order linear PDEs)**.** For a differential operator L in the form of eq. (2.6), define $(A(x))_{ij} = a_{ij}(x)$ and $b(x) = (b_1(x), \dots, b_n(x))^T$ for every $x \in \Omega$. L is called

- elliptic in x if all eigenvalues of $A(x)$ are nonzero and have the same sign,
- hyperbolic in x if all eigenvalues are nonzero, $n-1$ eigenvalues have the same sign and the remaining eigenvalue has the opposite sign,
- parabolic in x if one eigenvalue is zero, the remaining eigenvalues have the same sign and the $n \times (n+1)$ matrix $(A(x), b(x))$ has full rank.

The operator is said to be elliptic, hyperbolic or parabolic if the respective property holds for the entire domain Ω.

These classes of PDEs cover many practically relevant models including the applications in this thesis. For example, it is easy to verify that eq. (2.5) is an elliptic PDE, whereas the tsunami model in section 6.3 is based on a hyperbolic one. The classification itself plays an important role as it turns out that, while there certainly are similarities, each class requires its own theoretical and numerical treatment. The elliptic and, to a lesser extent, hyperbolic classes of PDEs will be the most prevalent in this work. For brevity we restrict ourselves to the elliptic class here and refer to [LeV02] for an extensive introduction to the hyperbolic class.

Towards a solution: Variational form and existence

One main goal of PDE theory is to investigate the existence and uniqueness of a solution. Theoretical treatment and numerical solution of PDEs are typically conducted in an abstract variational framework. We proceed to derive the variational form of Poisson's equation eq. (2.5), imposing boundary conditions that ensure uniqueness as we will later see. Again this serves as a specific example, and an analogous approach may be taken for many other PDEs. The PDE with Dirichlet boundary conditions (i.e. fixed value of solution prescribed) then reads

$$-\nabla \cdot (\kappa \nabla u) = f \qquad in \quad \Omega \tag{2.7}$$

$$u = 0 \qquad on \quad \partial\Omega. \tag{2.8}$$

First, we multiply both sides of eq. (2.7) by a test function v with $v|_{\partial\Omega} = 0$ and integrate.

$$\int_\Omega -\nabla \cdot (\kappa \nabla u) v dx = \int_\Omega f v dx.$$

Applying integration by parts on the left hand side and exploiting $v|_{\partial\Omega} = 0$, we arrive at

$$\underbrace{\int_\Omega (\kappa \nabla u) \cdot \nabla v dx}_{:= a(u,v)} = \underbrace{\int_\Omega f v dx}_{:= l(v)}.$$

This allows us to formulate an abstract variational PDE in terms of the bilinear form a and the linear form b:

$$\text{Find } u \in U \text{ such that:} \qquad a(u, v) = l(v) \qquad \forall v \in V. \tag{2.9}$$

Again, many other PDEs can be treated in an analogous way.

Clearly the existence of a solution as well as its properties strongly depend on the as of yet unspecified function spaces U and V. It turns out that the space $\mathcal{C}^1(\bar{\Omega})$ of differentiable functions on the closed domain with a norm based on a, the obvious choice for the PDE above, unfortunately leads to inconsistencies. Specifically, $\mathcal{C}^1(\bar{\Omega})$ is incomplete with respect to norms based on integrals. This issue can however be

mitigated by resorting to Sobolev spaces, in this case $\mathcal{H}_0^1(\Omega)$. Their introduction is outside the scope of this work and we refer to [BS07, Chapter II]. Existence and uniqueness is then guaranteed by the following theorem.

Theorem 1 (Lax-Milgram)**.** *Let $a : V \times V \to \mathbb{R}$ be a symmetric and continuous bilinear form and $l : V \to \mathbb{R}$ be a linear form.*

If a is coercive, i.e.

$$a(v, v) \geq \alpha \|v\|^2 \qquad \forall v \in V,$$

then eq. (2.9) *has a unique solution.*

Proof. See [BS07, Theorem 2.5]. □

Under mild assumptions on the coefficient κ, theorem 1 can be verified for Poisson's equation (see [Bas12, Section 6.2]). A full presentation of PDE theory exceeds the scope and we refer to [BS07] for a more rigorous treatment.

In addition to coercivity, the following property will play a role later:

Definition 2 (Stability)**.** A bilinear form $a : V \times V \to \mathbb{R}$ is called stable, if

$$|a(u, v)| \leq \gamma \|u\| \|v\| \qquad \forall u, v \in V,$$

and γ is called the stability constant.

Numerical solution

For elliptic problems, we can use a conforming FE method to reduce an approximate PDE solution to the solution of a linear system. Instead of trying to solve the variational problem eq. (2.9) for the entire function space V, we instead restrict ourselves to a finite dimensional subspace $V_h \subset V$.

$$\text{Find } u_h \in V_h \text{ such that:} \qquad a(u_h, v) = l(v) \qquad \forall v \in V_h. \tag{2.10}$$

For the subspace V_h, we assume a basis

$$\Phi_h = \{\phi_1^h, \dots, \phi_{N_h}^h\}.$$

The approximate solution u_h may now be expressed in terms of that basis:

$$u_h = \sum_{j=1}^{N_h} z_j \phi_j.$$

Inserting this basis representation into the variational problem eq. (2.10), it follows

$$a(u_h, v) = l(v) \qquad \forall v \in V_h \tag{2.11}$$

$$\iff \quad a(\sum_{j=1}^{N_h} z_j \phi_j, \phi_i) = l(\phi_i) \qquad \forall i = 1, \dots, N_h \tag{2.12}$$

$$\iff \quad \sum_{j=1}^{N_h} z_j \underbrace{a(\phi_j, \phi_i)}_{:= A_{ij}} = \underbrace{(\phi_i)}_{b_i} \qquad \forall i = 1, \dots, N_h \tag{2.13}$$

$$\iff \quad Az = b \tag{2.14}$$

The resulting linear system can now be solved with established methods. For an overview on iterative solvers for linear systems, refer to e.g. [Saa07]. Constructing computationally efficient solvers may however be challenging if the matrix A is ill-conditioned, and we will concern ourselves with that later in section 4.2.1.

This abstract procedure, known as the Galerkin method, works for any subspace V_h. But how to construct a suitable subspace? Especially for arbitrarily shaped domains Ω this is far from obvious. One widely used approach is to make use of a mesh on Ω.

Definition 3 (Mesh)**.** Given a domain $\Omega \subset \mathbb{R}^n$, a mesh on Ω is a finite set

$$\mathcal{T} = \{t_0, \dots, t_m\}$$

of bounded domains t_i, called elements, with Lipschitz boundary partitioning Ω in the sense of

$$\bar{\Omega} = \bigcup_{i=0}^{m} \bar{t}_m, \qquad t_i \cap t_j = \emptyset \quad \forall i \neq j.$$

Further, we define an element's size as $h(t) = \max_{x,y \in \bar{t}} \|x - y\|$ and the overall mesh size as $h = \max_{t \in \mathcal{T}} h(t)$.

In order to simplify the construction of a basis, a restriction to conforming and affine meshes is helpful.

Definition 4 (Affine mesh)**.** A mesh $\mathcal{T}$ is called affine if each element t_i is constructed from an affine map $\mu_{t_i} : \hat{\Omega} \to \Omega$ via

$$\bar{t}_i = \mu_{t_i}(\hat{\Omega}),$$

where $\hat{\Omega}$ is either the reference simplex

$$\hat{S}_n = \left\{ (x_1, \ldots, x_n) \in \mathbb{R}^n : 0 \leq x_i \leq 1, 0 \leq \sum_{i=1}^{n} x_i \leq 1 \right\}$$

or the reference cube

$$\hat{Q}_n = \{ (x_1, \ldots, x_n) \in \mathbb{R}^n : 0 \leq x_i \leq 1 \}.$$

Definition 5 (Conforming mesh)**.** An affine mesh $\mathcal{T}$ in $\mathbb{R}^n$ is called geometrically conforming if for any $t, t' \in \mathcal{T}$ it holds that $t \cap t'$ is either empty or a common face, i.e. a simplex or cube of dimension less than n.

Based on a conforming affine mesh it is now easy to construct a suitable basis.

Definition 6 (P_k and Q_k finite elements)**.** Let $\mathcal{T}$ be a simplicial conforming affine mesh on the domain Ω. Based on polynomials of degree k

$$\mathbb{P}_k^n = \{ u \in C^\infty(\mathbb{R}^n) : u(x) = \sum_{0 \leq |\alpha| \leq k} c_\alpha x^\alpha \},$$

the P_k space on $\mathcal{T}$ is defined as

$$P_k(\mathcal{T}) = \{u \in C^0(\bar{\Omega}) : u|_{\bar{t}} \in \mathbb{P}_k^n \quad \forall t \in \mathcal{T}.\}$$

Likewise, for a cuboid conforming affine mesh, the polynomial space

$$\mathbb{Q}_k^n = \{u \in C^\infty(\mathbb{R}^n) : u(x) = \sum_{0 \leq |\alpha|_\infty \leq k} c_\alpha x^\alpha\}$$

is used to define the corresponding Q_k space

$$Q_k(\mathcal{T}) = \{u \in C^0(\bar{\Omega}) : u|_{\bar{t}} \in \mathbb{Q}_k^n \quad \forall t \in \mathcal{T}.\}.$$

A basis for $P_k(\mathcal{T})$ and $Q_k(\mathcal{T})$ can now be constructed using Lagrange polynomials on the respective reference element and transforming them to the mesh elements [BS07][Chapter 5].

Finally, for a domain Ω with a corresponding conforming affine mesh $\mathcal{T}$, the basis of $P_k(\mathcal{T})$ or $Q_k(\mathcal{T})$ may be used to form a basis of the finite dimensional space V_h. The construction of the Galerkin method is therefore complete.

Note that there are numerous alternative ways to construct a FE type method. For example, discontinuous Galerkin methods allow for discontinuous function spaces and are frequently applied in flow problems [Kan07]. Another example is that the assumption $V_h \subset V$ may also be dropped in order to approximate boundaries that cannot be exactly matched by elements [CR72]. Those cases exceed the scope however.

It remains to show how the approximation error behaves that was introduced by substituting V by the lower-dimensional space V_h in eq. (2.10). A key result is the following.

Lemma 1 (Céa)**.** *Let a be a coercive bilinear form with coercivity constant α and stability constant C, i.e.*

$$|a(u, v)| \leq C\|u\|\|v\|.$$

Further assume $V_h \subset V$, let u denote the solution of the variational problem eq. (2.9) *and u_h the solution of the finite-dimensional problem eq.* (2.10)*. Then it holds*

$$\|u - u_h\|_a \leq \frac{C}{\alpha} \inf_{v \in V_h} \|u - v_h\|_a.$$

Proof. For the solutions u and u_h, we have by definition

$$\begin{aligned} a(u, v) &= l(v) \qquad \forall v \in V, \\ a(u_h, v) &= l(v) \qquad \forall v \in V_h. \end{aligned}$$

By subtraction, this implies

$$a(u - u_h, v) = 0 \qquad \forall V_h. \tag{2.15}$$

Now for any $v_h \in V_h$, we can conclude

$$\begin{aligned} \alpha \|u - u_h\|_a^2 &\leq a(u - u_h, u - u_h) \\ &= a(u - u_h, u - v_h + v_h - u_h) \\ &= a(u - u_h, u - v_h) + \underbrace{(u - u_h, v - u_h)}_{=0 \text{ due to eq. (2.15)}} \\ &\leq C \|u - u_h\|_a \|u - v_h\|_a \\ \iff \|u - u_h\|_a &\leq \frac{C}{\alpha} \|u - v_h\|_a \end{aligned}$$

Since the last inequality holds for all of V_h, it follows

$$\|u - u_h\|_a \leq \frac{C}{\alpha} \inf_{v \in V_h} \|u - v_h\|_a.$$

□

Lemma 1 shows that the approximation error introduced by solving the variational problem in V_h boils down to the question of how well the solution can be approximated in the space V_h. Since we construct V_h from piecewise polynomials, this question can be answered by interpolation results. Clearly the approximation behavior depends strongly on the regularity of the solution, the structure of the mesh as well as the polynomial basis used.

Definition 7 (Shape regularity)**.** A sequence of meshes $\{\mathcal{T}_\nu : \nu \in \mathbb{N}\}$ is called Shape regular if $\kappa_2 > 0$ indep. of ν s.t.

$$\forall \nu \in \mathbb{N}, \forall t \in \mathcal{T}_\nu : \frac{h_\nu(t)}{\rho_\nu(t)} \leq \kappa_2.$$

$\rho(t)$ diameter of largest ball that can be inscribed in t

Theorem 2 (P_k approximation result)**.** *Let $\{\mathcal{T}_\nu : \nu \in \mathbb{N}\}$ be a sequence of affine and shape regular simplicial meshes on Ω with $h_\nu \to 0$. Then for the interpolation operator $\mathcal{I}_\nu : H^k(\Omega) \to P_{k-1}(\mathcal{T}_\nu)$ the following error bound holds:*

$$\|u - \mathcal{I}_\nu u\|_{m,\Omega} \leq C h^{k-m} |u|_{k,\Omega} \qquad \forall u \in H^k(\Omega), 0 \leq m \leq m,$$

where the constant C depends only on the shape regularity constant and space dimension of Ω.

Proof. See [Bas12, Theorem 8.10]. □

In this result, $H^k(\Omega)$ is the Sobolev space of order k on the domain Ω with the associated seminorm $|\cdot|_{k,\Omega}$, which becomes a true norm on the left hand side since $(u - \mathcal{I}_\nu)|_{\partial\Omega} = 0$. Again for the purposes of this thesis it is enough to think of the Sobolev space H^k as a completion of C^k and we refer to [BS07, Chapter II] for details.

Combining theorem 2 with lemma 1 and some more technical intermediate steps left out here for brevity, an a priori error estimate for the FE method can be shown ([Bas12, Theorem 8.16]):

$$\|u - u_h\|_{1,\Omega} \leq C h^{k-1} \|f\|_{k-2,\Omega}.$$

This result is of course restricted to the assumptions made above. However, it is very representative of numerous numerical PDE methods in the sense that their respective analysis typically provides a priori estimates of that form. The convergence behavior depending on polynomial degree and mesh width to a certain power can later be used to satisfy the assumptions on accuracy per cost required by multilevel UQ methods in chapter 3. Simply changing mesh width across levels is therefore often enough to generate suitable model hierarchies. This and some more advanced approaches will be discussed in chapter 4.

2.2 Basic Concepts of Probability Theory

This section introduces the underlying probability theory notions and tools required later to formulate and treat probabilistic models. In particular, uncertain parameters and measurements will be expressed using random variables, and so will be corresponding model predictions. The numerical methods employed will then aim to efficiently approximate the distributions of the unknown random variables, or statistical moments derived therefrom.

Anyone familiar with probability theory, specifically Markov chains, can easily skip this section, as it is simply a review of well-established probability theory. We largely follow the presentation in [SZ21] and also draw from [Sul15]. For readers new to probability, this should give sufficient insight to follow subsequent chapters.

As a very first step, we begin to define the most basic component of probability theory: A measurable space. Its purpose is essentially to define a set of possible random events, to which we later assign individual probabilities. In case of regular dice, we might have six possible events. In case of modeling an unknown physical parameter, we might actually consider the entirety of the real numbers $\mathbb{R}$ as the set of possible events.

Definition 8 (Sample space and σ-algebra)**.** We denote a set of underlying random events by $\Omega \neq \emptyset$. Then, we call a subset of its power set $\mathcal{F} \subset Pot(\Omega)$ a σ-Algebra over Ω if it holds

1. $\Omega \in \mathcal{F}$,
2. $A \in \mathcal{F} \Rightarrow A^{\complement} \in \mathcal{F}$,
3. $A_i \in A \quad \forall i \in \mathbb{N} \Rightarrow \bigcup_{i\in\mathbb{N}} A_i \in \mathcal{F}$.

Together, we call $(\Omega, \mathcal{F})$ a measurable space and any set $A \in \mathcal{F}$ a measurable set in that space.

Now that we have a structure to hold basic and 'combined' events, we can proceed to assign probabilities to them. To that end, we introduce measures on measurable spaces.

Definition 9 (Measure and probability measure)**.** Given a measurable space $(\Omega, \mathcal{F})$, we call a function $\mu : \mathcal{F} \to \mathbb{R}_0^+$ a measure if

1. $\mu(\emptyset) = 0$ and

2. for disjoint sets $A_i \in \mathcal{F}$, $i \in \mathbb{N}$, it holds

$$\mu(\bigcup_{i\in\mathbb{N}} A_i) = \sum_{i\in\mathbb{N}} \mu(A_i).$$

Further, in case $\mu(\Omega) = 1$, we call μ a probability measure.

We will further need the notion of a property holding almost everywhere with respect to a measure.

Definition 10 (Null set and properties holding almost-everywhere)**.** For a measure μ, we call $N \in \mathcal{F}$ a null set if $\mu(N) = 0$.

If there is a null set A such that a certain property holds for all elements of $A^{\complement}$, we say this property holds μ-almost everywhere.

We will often work with measurable spaces that have a specific associated measure, which leads us to the following definition.

Definition 11 (Measure space and probability space)**.** When extending a measurable space $(\Omega, \mathcal{F})$ by a measure μ defined on it a measure space and denote it by $(\Omega, \mathcal{F}, \mu)$.

In case of a probability measure $\mathbb{P}$, we refer to the measure space $(\Omega, \mathcal{F}, \mathbb{P})$ as a probability space.

When mapping random events to another space in a suitable way, as might be the case when mapping stochastic parameters to stochastic model predictions, subsets of the image could be measured by their corresponding preimages. To that end, we first define measurable functions.

Definition 12 (Measurability of functions)**.** A function $f : \Omega_1 \to \Omega_2$ mapping between two measurable spaces $(\Omega_1, \mathcal{F}_1)$ and $(\Omega_2, \mathcal{F}_2)$ is called $\mathcal{F}_1/\mathcal{F}_2$-measurable if $f^{-1}(A_2) \in \mathcal{F}_1$ holds for all $A_2 \in \mathcal{F}_2$.

Now we can proceed to define the measure implied on the image space of such a function.

Definition 13 (Pushforward measure)**.** For a measurable function $f : \Omega_1 \to \Omega_2$ defined on a probability space $(\Omega_1, \mathcal{F}_1, \mu)$ and a measurable space $(\Omega_2, \mathcal{F}_2)$, we define the pushforward measure

$$f_*(A_2) \coloneqq \mu(f^{-1}(A_2)) \quad \forall A_2 \in \mathcal{F}_2.$$

It is easy to show that the pushforward f_* is actually a measure on $(\Omega_2, \mathcal{F}_2)$.

For most of the following work, we will make use of the following special case.

Definition 14 (Random variable)**.** A measurable function $X : \Omega \to V$, where V is a Banach space, defined on measurable spaces $(\Omega, \mathcal{F})$ and $(V, \mathcal{F}_2)$ is called a V-valued Random Variable (RV).

Note that in the following, we often implicitly assume $\mathcal{F}$ and $\mathcal{F}_2$ to be suitably defined and focus entirely on the definition of the RV as a mapping.

Since a RV is itself a measurable function, it also implies its own pushforward measure when defined on a probability space.

Definition 15 (Distribution of a RV)**.** Let X a RV defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We call the pushforward measure $\mathbb{P}_X \coloneqq X_*\mathbb{P}$ the RV's distribution.

In the following, we write $X \sim \mu$ to indicate that the RV X has distribution μ.

While the goal of many methods introduced in the following sections is to determine the entire distribution of parameters or model predictions, higher-level statistics (expectation values, variances etc.) of these distributions are often of interest. Aside from giving an easy to grasp understanding of a potentially high-dimensional resulting distribution, they also play a key role in analyzing the accuracy and efficiency of those methods. Since they are typically defined as integrals, we first have to introduce the Bochner integral as a generalization of the Lebesgue integral (for a more detailed introduction, refer to e.g. [Bau01]).

Definition 16 (μ-simple functions)**.** We call a measure space $(\Omega, \mathcal{F}, \mu)$ (or the measure μ itself) σ-finite if there exists a countable set $(A_i)_{i\in\mathbb{N}} \in \mathcal{F}$ such that

$$\bigcup_{i\in\mathbb{N}} A_i = \Omega \quad and \quad \mu(A_i) < \infty \quad \forall i \in \mathbb{N}.$$

A function $f : \Omega \to V$ defined on such a σ-finite measure space is called μ-simple if we can represent it based on finitely many event sets, that is

$$f = \sum_{i=1}^{n} \mathbb{I}_{A_i} v_i,$$

where $v_i \in V$, $A_i \in \mathcal{F}$ and $\mu(A_i) < \infty$ holds for all $1 \leq i \leq n$.

Definition 17 (Bochner integral)**.** A function $f : \Omega \to V$ defined on a σ-finite measure space $(\Omega, \mathcal{F}, \mu)$ and a Banach space V is called μ-Bochner integrable if

1. a sequence of μ-simple functions $f_n = \sum_{i=1}^{N} \mathbb{I}_{A_{n,i}} v_{n,i}$ exists such that $lim_{n\to\infty} f_n = f$ μ-almost everywhere and
2. $$lim_{n\to\infty} \int_\Omega \|f(\omega) - f_n(\omega)\| d\mu(\omega) = 0.$$

In that case, the Bochner integral is defined as

$$\int_\Omega f(\omega) d\mu(\omega) \coloneqq lim_{n\to\infty} \sum_{i=1}^{n} \mu(A_{n,i}) v_{n,i}.$$

Based on that, we can introduce the expectation value of a RV.

Definition 18 (Expectation value)**.** For a RV $X : \Omega \to V$, we define its expectation as

$$\mathbb{E}[X] \coloneqq \int_\Omega X(\omega) d\mathbb{P}(\omega)$$

if the integral exists.

We can now proceed to introduce the covariance operator, which can be seen as indicating the spread of and relations between different components of a RV.

Definition 19 (Covariance operator)**.** Given two RVs $X : \Omega \to \mathbb{R}^n$ and $Y : \Omega \to \mathbb{R}^m$ defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, we define the covariance operator $cov(X, Y) : \mathbb{R}^m \to \mathbb{R}^n$ by

$$\langle v, cov(X, Y) w \rangle = \int_\Omega \langle X - \mathbb{E}[X], v \rangle \langle Y - \mathbb{E}[Y], w \rangle d\mathbb{P}$$

provided the integral exists for any preimage of the operator.

Definition 20 (Variance and covariance matrix)**.** In case of a RV $X : \Omega \to \mathbb{R}$ defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, we define the variance of X as

$$\mathbb{V}(X) \coloneqq \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2.$$

For RVs $X : \Omega \to \mathbb{R}^n$ and $X : \Omega \to \mathbb{R}^m$ this can be generalized to the algebraic representation of the covariance operator, namely the covariance matrix

$$C_{ij} = \mathbb{E}[(X_i - \mathbb{E}[X_i])(Y_j - \mathbb{E}[Y_j])] \quad \in \mathbb{R}^{n \times m}.$$

Note that for a single RV X in $\mathbb{R}^1$ it holds $C_{11} = \mathbb{V}[X]$.

When treating inverse UQ problems, we will make use of Markov Chain Monte Carlo (MCMC) type methods. Therefore, we introduce the underlying notion of Markov chains and related concepts here, following [Beh00] and [Bro+11]. First of all, a Markov chain is a special case of a stochastic process, which is defined as follows.

Definition 21 (Stochastic process)**.** Given a set S, an S-valued stochastic process is a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ together with random variables $X_k : \Omega \to S$, $k \in \mathbb{N}_0$.

A Markov chain is now a stochastic process where a random variable X_k may only depend on its predecessor X_{k-1}.

Definition 22 (Markov chain)**.** An S-valued stochastic process X_0, X_1, ... is called a Markov chain if for every $k \in \mathbb{N}$ and arbitrary $i_0, ..., i_{k-1}, j \in S$ it holds

$$\mathbb{P}(X_k = j | X_0 = i_0, ..., X_{k-1} = i_{k-1}) = \mathbb{P}(X_k = j | X_{k-1} = i_{k-1}).$$

It is said to have stationary transition probabilities if the distribution of X_{k+1} given X_k does not depend on k.

Note that while Markov chains can, in case of finite S, be defined purely in terms of transition probabilities, that more elementary definition is ultimately equivalent to a stochastic process as above (see [Beh00][Theorem 1.4]). For brevity, we therefore directly work with the definition based on stochastic processes.

The stationarity property will play a key role in MCMC methods, and it is defined as follows.

Definition 23 (Stationarity of stochastic processes)**.** A stochastic process X_0, X_1, ... is called stationary if for every $k \in \mathbb{N}$ the distribution of the k-tuple

$$(X_{n+1}, ..., X_{n+k})$$

does not depend on n.

Another property relevant to MCMC is autocorrelation. In MCMC we are indirectly generating samples from a target distribution using a Markov chain. Low autocorrelation means we achieve near independent samples, leading to high quality estimates with few samples, while high autocorrelation essentially implies redundancy in our generated samples. Therefore autocorrelation is a useful measure regarding efficiency of MCMC methods.

Definition 24 (Autocovariance, effective sample size)**.** For a stationary stochastic process X_0, X_1, ..., we define its lag-k autocovariance as

$$\gamma_k = cov(X_i, X_{i+k}),$$

the autocorrelation function as $k \mapsto \gamma_k$ and. In case the γ_k have one component, we additionally define the autocorrelation function as $k \mapsto \gamma_k/\gamma_0$.

We further define the integrated autocorrelation time as

$$\tau = 1 + 2\sum_{k=1}^{\infty} corr(X_0, X_k), \tag{2.16}$$

where the correlation in turn is defined as $corr(X_0, X_k) := \frac{cov(X_0, X_k)}{\sqrt{\mathbb{V}[X_0]}\sqrt{\mathbb{V}[X_k]}}$.

In case only a finite subset $\{X_0, ..., X_N\}$ of the process is accessible, we refer to the fraction N/τ as the Effective Sample Size (ESS) of that subset.

The notion of effective sample size later becomes crucial when gauging the quality of an MCMC estimator consisting of a truncated stationary Markov chain, since less correlated samples generated by MCMC lead to a lower approximation error. Determining the ESS in practice is not a trivial task, and in practical applications we use the implementation of the approach in [Wol04] provided by the MIT Uncertainty Quantification Library (MUQ).

2.3 Uncertainty Quantification Problems

There is a wide variety of theoretically interesting and practically relevant UQ problems. They mainly come down to introducing uncertainty to parts of an otherwise deterministic model and investigating what the effect on the information we seek from the model is.

Typically, UQ problems fall into two categories:

- Forward propagation refers to investigating the distribution of model predictions for uncertain parameters given in terms of a specific distribution.
- Inverse problems seek to find the underlying parameters explaining given measurements. Those measurements often come from real-world observations, but for testing purposes may also be generated artificially. Since those measurements are typically affected by uncertainty and the model itself may not be uniquely invertible, the result is given in terms of a parameter distribution.

2.3.1 Forward problems

Forward UQ problems can be viewed as simple stochastic extensions to deterministic models. A deterministic model F maps a specific parameter vector $\theta \in \mathbb{R}^m$ onto a specific model prediction $F(\theta) \in \mathbb{R}^n$ (see fig. 2.1). We consider finite dimensional parameters and model predictions in the following, even though infinite dimensional parameter spaces may be treated in a similar way (see nonparametric UQ problems [Hjo+10]).

We also introduce the notion of a Quantity of Interest (QOI) here. Quite often, the model prediction itself is not the practically relevant quantity; instead, values derived from it are of interest. For example, a weather forecast system internally produces predictions on temperature, pressure, humidity etc. for the entire simulation domain. The interesting outcome, however, might be whether it is likely to rain today in Heidelberg. In the setting of forward models, disambiguating QOI and model prediction seems redundant, as the model prediction could simply be defined as our QOI. However, in inverse UQ problems, the QOI plays a crucial role by itself, so we already add it for consistency.

In a setting where the specific parameter θ is affected by uncertainty, we may model this uncertainty in terms of a known parameter distribution π_0 (see fig. 2.2). We now consider θ a RV with $\theta \sim \pi_0$. For example, if the parameter can be directly measured

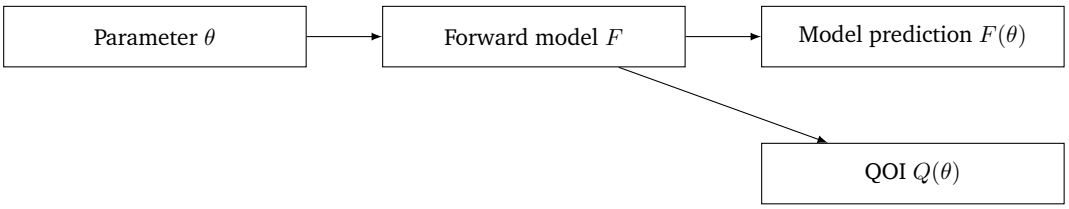


Fig. 2.1: Abstract deterministic model mapping parameters to model predictions.

experimentally, the distribution π_0 could be a Gaussian distribution centered on the measured value, with a variance matching the accuracy of the measurement apparatus. Now we do not have a single model prediction any more, but instead an entire distribution of predictions resulting from the underlying parameter distribution. More specifically, the result is a pushforward of the parameter distribution by our model map. Likewise, the QOI prediction becomes a pushforward distribution as well.

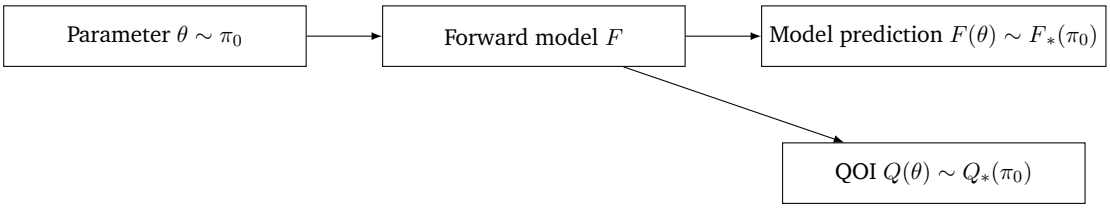


Fig. 2.2: Extension to stochastic model based on a known parameter distribution.

2.3.2 Inverse problems

Forward problems as introduced above require full knowledge of the underlying parameter or, in the stochastic case, the respective parameter distribution. There are many practical applications however, where the underlying parameter is entirely unknown, possibly up to some vague a priori information. Instead, those parameters would have to be inferred through indirect observation.

If a unique inverse F^{-1} of our forward model was available, computing the parameter leading to a certain real-world measurement y was as simple as evaluating the inverse model (see fig. 2.3).

In analogy to the forward setting, we could determine uncertainty in the underlying parameter by simply applying the inverse model to a measurement distribution encoding uncertainty in our measurement process (see fig. 2.4).

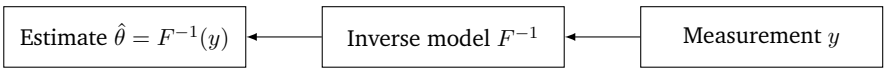


Fig. 2.3: Abstract deterministic inverse model mapping measurements to inferred parameters.

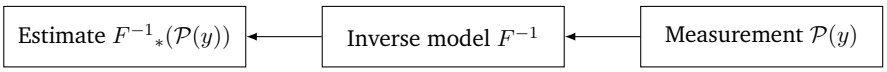


Fig. 2.4: Abstract inverse model mapping distribution of measurements to inferred parameter distribution.

In reality, however, the inverse of our model map F is often computationally intractable or simply not well-defined. A simple and extreme example can be found in physical processes like heat diffusion: Once equilibrium is reached, the initial heat distribution cannot be recovered from measurements (except for the total amount of energy). Even in more promising settings, the inverse is often not unique.

As a result, simply treating inverse problems in analogy to forward problems by switching out the model map for its inverse is not sufficient, and we need to turn to a more sophisticated approach. We pose the following requirements:

- Only the model map F should be evaluated, not its inverse, and
- no additional knowledge about the model (e.g. derivatives) should be needed.

An established approach to avoid the inverse model map is the Bayesian framework. The basic idea is to exploit Bayes' theorem [Dod08] in order to express the posterior density, which represents the probability of parameters explaining a given measurement, in terms of a likelihood density and a prior density.

Definition 25 (Posterior density)**.** For given likelihood $\mathcal{L}$, prior π_0, we call

$$\pi(\theta) := \frac{\mathcal{L}(y|\theta)\pi_0(\theta)}{\mathcal{P}(y)}$$

the posterior density of θ given fixed measurements y, and ν the corresponding distribution.

The likelihood $\mathcal{L}(y|\theta)$ represents the probability of a given parameter leading to the observed measurement. Typically, this requires evaluating the model for the parameter θ and comparing the outcome to the given measurement y. When $F(\theta)$ is

close to y, in whatever metric makes sense in the context, the likelihood should have a high value. In the opposite case, the likelihood should be low. A typical choice for the likelihood of y given θ is a Gaussian distribution $\mathcal{N}(F(\theta), \Sigma_F)$, where Σ_F encodes the accuracy of the measurements in terms of a covariance matrix. We also need to define a prior density π_0. Since this is not informed by measurements, it encodes a priori knowledge about parameters. For example, a model expert might consider a certain parameter range more plausible than others. The final component $\mathcal{P}(y)$ poses an issue, since the unconditioned probability of measurement values is unobtainable in many applications and might only be derived from the unknown underlying parameter distribution. However, with respect to the parameter θ this is a constant, and we will later introduce methods that can ignore this scaling factor.

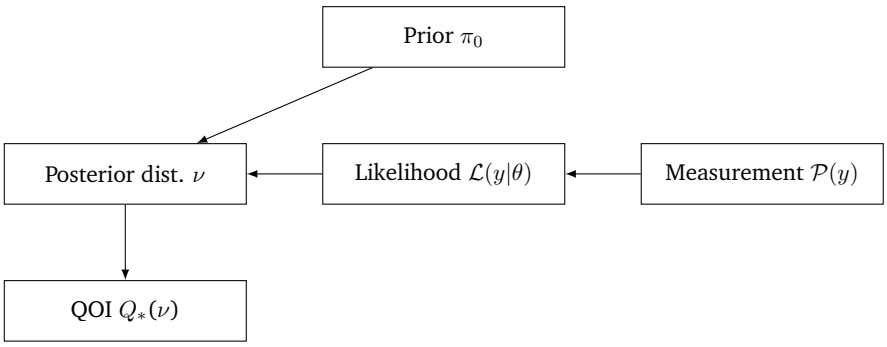


Fig. 2.5: Abstract deterministic inverse model mapping a measurement distribution to an inferred parameter distribution using Bayes' framework.

Compared to our naive approach above (fig. 2.4), we now have a slightly more complex framework (see fig. 2.5). We have, however, gained the ability to infer parameters from uncertain measurements through a potentially non-invertible forward model.

2.4 Basic Numerical Uncertainty Quantification Algorithms

Now that we have introduced interesting UQ problems in section 2.3, we obviously arrive at the question: How can we solve them? More specifically, how can we obtain the pushforward or posterior distributions respectively?

Ideally, we would like to have an exact analytical solution. In some very simple cases, a solution can be computed on paper. For example, if we start from an

uncertain parameter with Gaussian distribution $\theta \sim \mathcal{N}(0,1)$ and have a simple affine transformation $F(x) = x + 1$ as our forward model, the model prediction again has a Gaussian distribution $F(\theta) \sim \mathcal{N}(1,1)$.

For more complex models, however, such a calculation will be outright impossible once the model itself can only be evaluated numerically. As a result, we need to employ numerical approximations to the stochastic problem as well.

This is analogous to many other numerical fields: For example, some Ordinary Differential Equation (ODE)s and PDEs can be solved on paper; once the equations becomes more intricate, as is usually the case when real-world data enters, one has to resort to a numerical solution. The analytical problem is discretized in a way that allows it to be solved algorithmically in finite time.

The main goal of numerical method development is to improve approximation quality in relation to computational effort. Numerical results can demonstrate practical applicability, uncover further challenges and show computational efficiency. Theoretical treatment proves the methods' mathematical correctness and reliability, and ideally provides insights to support future method development.

Following that structure, we proceed to introduce basic numerical approximations to the problems above. While far from computationally efficient, they already provide valid solutions. They also form the underlying building blocks of the more advanced methods in chapter 3, which mainly set themselves apart regarding efficiency, and correspondingly the scale of problems they can treat effectively.

A very accessible introduction to these basic methods can be found in [McC18], and this section is in part inspired by this source.

2.4.1 Illustrative Model

In order to illustrate the UQ methods introduced in the following sections, we first define a very simple deterministic model problem. This ODE models the trajectory of a basketball thrown by a 1.8m tall player, whose goal is to hit a standard dimensions basket (3.05m tall, 46cm diameter) at a distance of 4m. As our model's parameter, we define the initial velocity $\theta = v^0 = \begin{pmatrix} v_x^0 \\ v_y^0 \end{pmatrix}$.

The trajectory can then be defined as the system of ODEs

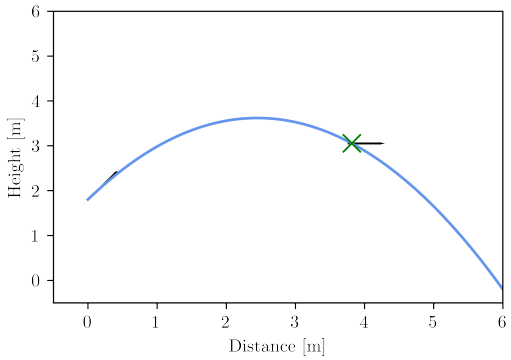


Fig. 2.6: Example trajectory of the model problem for an initial velocity of $7.2\frac{m}{s}$ at an angle of 56 degrees. The basket is indicated as a black line, and the green cross represents the point where the trajectory hits the basket.

$$x'(t) = v_x^0$$
$$y''(t) = -g$$

with initial values

$$x(0) = 0$$
$$y(0) = 1.8m$$
$$y'(0) = v_y^0.$$

As our QOI, we define whether a given trajectory reaches the basket, i.e.

$$Q(\theta) = \begin{cases} 1 & \text{trajectory hits basket} \\ 0 & \text{trajectory misses basket} \end{cases}$$

Note that for numerical calculations, this system can easily be transformed into one with first order derivatives only.

For the UQ methods introduced later, the cost for achieving a certain accuracy in the numerical solution will be crucial. We will assume that the above ODE is solved with a simple explicit Euler's method and equidistant step size h. From this method, since we have a sufficiently smooth right hand side of the ODE, we may expect

a convergence rate of $\mathcal{O}(h)$ to the true solution. For a thorough introduction to numerical methods for ODEs, we refer to the established literature, e.g. [Hol07].

2.4.2 Monte Carlo Methods

In this section we give a rudimentary, yet already feasible answer to the question: How to numerically solve forward UQ problems?

As detailed in section 2.3.1, we seek the distribution of model predictions

$$F(\theta) \sim F_* \pi_0$$

produced by a model F with uncertain input modeled as a RV $\theta \sim \pi_0$.

The approach we take is a rather intuitive one: We draw independent samples $\theta_1, ..., \theta_N$ from our input distribution π_0. For a sufficiently large number of samples N, statistics of these samples will represent those of the actual distribution well. In order to estimate the expected value, for example, we can take the following approximation:

$$\mathbb{E}[\theta] \approx \frac{1}{N} \sum_{i=1}^{N} \theta_i.$$

The law of large numbers immediately guarantees that, since our samples are Independent and identically distributed (i.i.d.) by construction, this approximation is exact for $N \to \infty$. Since we know the exact distribution π_0 (after all it is part of the design of our forward UQ problem), we can typically use established pseudo random generators to generate samples in practice.

Now that we have a discrete representation of our input distribution, we can easily compute a corresponding discrete representation of our pushforward distribution: All it takes is applying the model map to each sample of the input distribution, namely

$$F(\theta_1), ..., F(\theta_N),$$

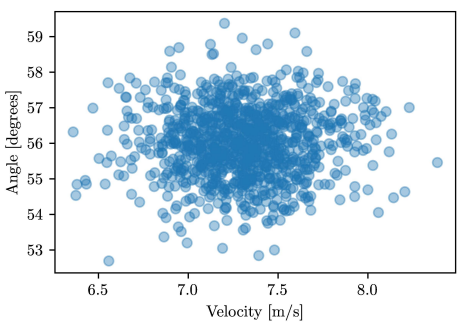
which can be seen as applying the pushforward. Since those samples are i.i.d. samples from $F_* \pi_0$, we can continue the example above and again approximate the mean by

$$\mathbb{E}[F(\theta)] \approx \widehat{F}_N^{MC} := \frac{1}{N}\sum_{i=1}^{N} F(\theta_i),$$

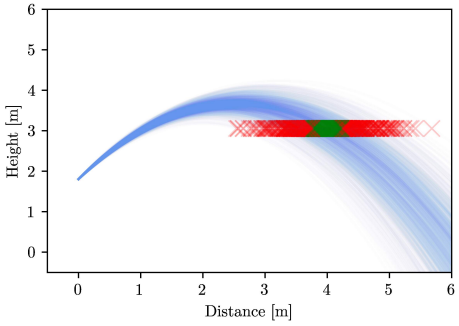
or compute any other statistics. This approach is known as the Monte Carlo (MC) method. From the definition of this method it is easy to see that we only need to evaluate the forward model, and a finite number of evaluations is enough to deliver the approximation we seek. As a result, the method can easily be implemented in practice, for example using numerical forward model solutions.

Application to model

Let us now put this first method to use on our model problem from section 2.4.1: First, we can extend this deterministic model to a forward UQ problem by formulating a prior distribution. A somewhat realistic choice is $\theta \sim \mathcal{N}(\mu, \sigma)$ where $\mu = (7.2, 56)^T$ and $\sigma = \begin{pmatrix} .3 & 0 \\ 0 & 1.0 \end{pmatrix}$. This models a variation in throwing angle and velocity, as would be expected in a real-world scenario with a human basketball player. The question we are going to answer is: How likely is our slightly unreliable basketball player to hit the basket?



(a) Samples θ_i from prior distribution.

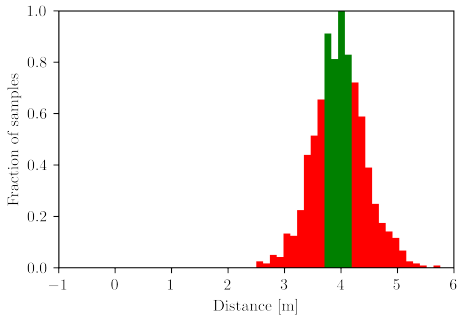


(b) Model states computed for the above set of parameters. Hits are indicated by green, misses by red markers.

Fig. 2.7: Monte Carlo prior sampling and resulting model states.

Applying a Monte Carlo sampler to the model problem is fairly easy: We draw a set of random parameters $\{\theta_i\}_{i=1}^N$ where $\theta_i \in \mathbb{R}^2$ from the two-dimensional Gaussian prior distribution. For this problem, a relatively small number of samples $N = 1000$ should be sufficient. Figure 2.7a shows such a set of samples. Now that we have a discrete representation of our distribution, we can apply the model to each parameter. Note that each model evaluation is now technically equivalent to

evaluating a deterministic model for a single parameter, and as such neither the mathematical model nor the implementation would have to be adapted.



(a) The distance component x of all model evaluations at the height of the basket.

Hits	Samples	Hit ratio
400	1000	0.4

(b) Hit ratio evaluation via $\mathbb{E}[Q(\theta)]$.

Fig. 2.8: QOI as computed via Monte Carlo method.

After applying the model to those prior samples, the resulting states $F(\theta_i)$ (in this case, ODE solutions) are shown in fig. 2.7b. Since we are not actually interested in the entire trajectories, we can now proceed with post-processing on those states, particularly calculating our QOI samples. Figure 2.8a shows the x component of the ODE solution at the height of the basket. From that, it is easy to compute the desired value, namely the probability hitting the basket: It is enough to check what fraction of trajectories reaches the basket relative to the total number of samples. The results are shown in fig. 2.8b.

Efficiency considerations

Clearly, this method may require a large number of samples. But how does its cost scale? The following quick calculation is enough to give us a rough idea about that.

A frequently used quality indication of an estimator is its Mean square error (MSE) which, assuming an unbiased estimator, is equivalent to the estimator's variance.

$$\mathbb{E}[(\widehat{F}_N^{MC} - \mathbb{E}[F(\theta)])^2] = \mathbb{V}[\widehat{F}_N^{MC}].$$

Further, we can exploit that our estimator is a sum over i.i.d. samples, and thus it holds

$$\mathbb{V}[\widehat{F}_N^{MC}] = \mathbb{V}[\frac{1}{N}\sum_{i=1}^{N} F(\theta_i)] = \frac{1}{N}\mathbb{V}[F(\theta)].$$

We can now compute (up to a constant) the number of samples needed to achieve a MSE below ϵ^2:

$$\begin{aligned}\mathbb{V}[\widehat{F}_N^{MC}] &= \frac{1}{N}\mathbb{V}[F(\theta)] \leq \epsilon^2 \\ \implies N &\geq \frac{\mathbb{V}[F(\theta)]}{\epsilon^2} = \mathcal{O}(\epsilon^{-2}).\end{aligned}$$

Finally, assuming that all forward model evaluations are of equal cost $\mathcal{C}_F$, we arrive at

$$Cost(\widehat{F}_N^{MC}) = \mathcal{C}_F N = \mathcal{O}(\epsilon^{-2}).$$

This cost bound holds true if the forward model F is directly accessible. In practice, this is often not the case: As we have already seen in section 2.1, often only numerical discretizations of the model are available. These obviously introduce an error in addition to the sampling error discussed above, since we can only sample from an approximate pushforward distribution induced by the discretized forward model. Controlling the discretization error usually implies that model cost $\mathcal{C}_F$ also scales with some power of ϵ.

For brevity, we only fully consider model cost per accuracy in the cost analysis of MCMC in section 2.4.3. It shall be noted, however, that a cost estimate analogous to eq. (2.19) can be shown for MC as well:

$$Cost(\widehat{F}_N^{MC}) = \mathcal{O}(\epsilon^{-2-\frac{\gamma}{\alpha}}),$$

where α is the convergence rate of the expected value of the pushforward depending on model discretization level and γ is the corresponding rate of cost increase.

Solving the illustrative model from section 2.4.1 with Euler's method, we can expect linear increase in cost ($\gamma = 1$) for linear error reduction ($\alpha = 1$), and we therefore arrive at $Cost(\widehat{F}_N^{MC}) = \mathcal{O}(\epsilon^{-3})$.

Cost increasing cubically for given desired accuracy is obviously not ideal, particularly when working with large models. In chapter 3 we will present more advanced methods whose main goal is improving that convergence rate.

2.4.3 Markov Chain Monte Carlo Methods

As we have seen above, a simple Monte Carlo method directly applied to uncertain parameters allows us to numerically solve forward UQ problems. But can we extend this concept to inverse problems as well?

In section 2.3.2 we already discussed that, if we had an inverse of the model map available, we could interpret inverse problems as forward problems based on F^{-1}. However, since this is very often not available in practice, we had to introduce the Bayesian framework. This lead us to a posterior density

$$\pi := \frac{\mathcal{L}(y|\theta)\pi_0(\theta)}{\mathcal{P}(y)} \propto \mathcal{L}(y|\theta)\pi_0(\theta)$$

where only our forward model enters as part of the likelihood.

As in Monte Carlo, we would like to represent this posterior by a finite number of samples. How can we draw samples from this distribution? Directly sampling from it would require very restrictive assumptions on the model, and without knowledge of the true parameter distribution or large amounts of measurements the scaling factor $\mathcal{P}(y)$ is entirely inaccessible.

An established approach to solve these issues are MCMC methods. The main idea is to generate a Markov chain over the parameter space and ensure that its stationary distribution matches the posterior. This can be ensured by essentially producing a random walk which is weighted to reside in high-probability regions, but also occasionally step into lower-probability regions. Typically, weighting is achieved by evaluating the posterior for each step. If we now only take a finite number of steps in the Markov chain, we obtain a (hopefully) good approximation of the posterior while only evaluating it for a finite number of specific parameters.

There is a wide variety of methods of this type, balancing computational efficiency against required model information. For example, some more advanced methods may require derivatives of the model map. At their core, however, they are mostly variations of the same underlying scheme. A very simple one, and in fact historically one of the earliest, is Metropolis-Hastings Markov Chain Monte Carlo (MHMCMC) [Met+53; Has70], which we now introduce as a representative example.

For a more detailed discussion of advanced MCMC methods and alternatives, refer to chapter 3.

Metropolis Hastings Markov Chain Monte Carlo

In MHMCMC, we generate such a Markov chain approximating a target distribution π by the following algorithm. From a starting point, we generate a sequence of samples $\{\theta^i\}_{i=0}^N$ through what is essentially a random walk across the parameter space. A new step in the vicinity of the current one is proposed through a proposal distribution q. Whether we take that step, however, is something we decide depending on whether our target distribution has a higher value there. In order to also cover low-probability regions, albeit with a correspondingly lower number of samples, we sometimes admit low-probability proposals as well. The exact procedure is detailed in algorithm 1 and can be shown to indeed yield the desired stationary distribution π.

Algorithm 1: MHMCMC

Result: Markov chain $\{\theta^i\}_{i=0}^N$.
Initialize the Markov chain with an arbitrary starting point $\theta^0 \in \mathbb{R}^m$.
for *sample* $n = 0, \dots, N-1$ **do**

- Draw a correlated proposal θ' from a proposal distribution $q(\theta'|\theta^n)$.
- For this proposal, compute an acceptance probability

$$\alpha(\theta'|\theta^n) = \min\left\{1, \frac{\pi(\theta')q(\theta^n|\theta')}{\pi(\theta^n)q(\theta'|\theta^n)}\right\}.$$

- With probability α, accept the proposal θ' to form the next step of the chain and set $\theta^{n+1} = \theta'$. Otherwise, the chain should remain at the previous parameter value, i.e. $\theta^{n+1} = \theta^n$.

end

Classically, one would choose the proposal distribution q as a Gaussian distribution centered around the previous step, and in fact this comes very close to the intuitive interpretation as a weighted random walk. Later we will see that more advanced proposal distribution can offer significant improvements in convergence.

Application to model

In order to illustrate the concept of inverse problems and the aforementioned MHMCMC algorithm, we extend our deterministic model problem from section 2.4.1 to an inverse UQ problem. Since our goal is to infer underlying "true" parameters from indirect observations, we first have to define what types of measurements we will consider.

We suppose that we have a camera taking pictures of the ball when it reaches a distance of 1.5m and 2.5m from the basketball player respectively, and that we can observe the ball's height at these points - up to a certain accuracy, of course. In order to test the method, we assume the underlying "true" parameter to be $\begin{pmatrix} 7.2 \\ 56 \end{pmatrix}$ Figure 2.9 shows the corresponding model output as well as measurement points along the trajectory.

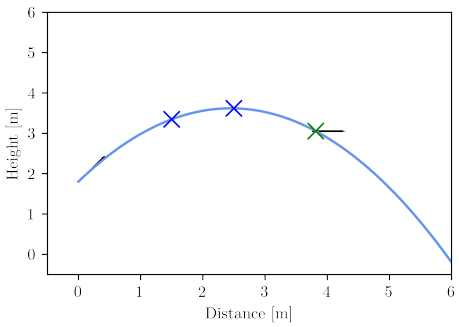


Fig. 2.9: "True" trajectory with height measurements (blue crosses).

Of course that "true" parameter would be unknown in a realistic application. So how can we infer the "true" throwing parameters just from resulting measurements? And, most importantly, how can we determine our QOI, namely the probability of hitting the basket given those measurements?

We follow the recipe from section 2.3.2 and formulate a Bayesian inverse problem. For our prior, we choose

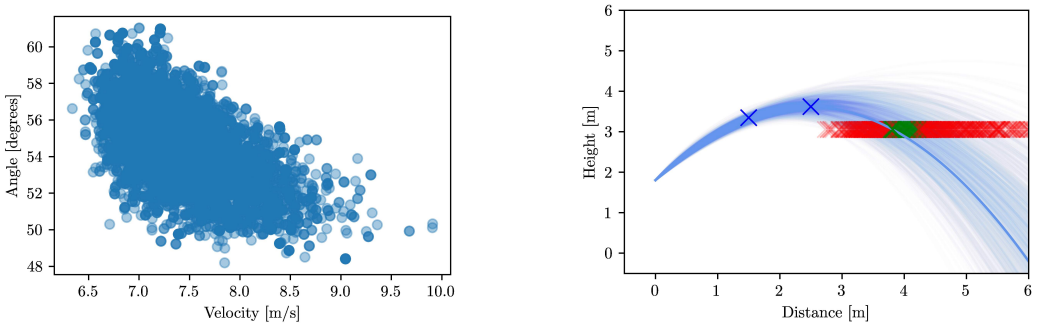
$$\pi_0 := \mathcal{N}\left(\begin{pmatrix} 7 \\ 55 \end{pmatrix}, \begin{pmatrix} 2 & 0 \\ 0 & 5 \end{pmatrix}\right),$$

which should cover the range of throws to be expected in a realistic scenario. Clearly this a priori knowledge can only give us a rough idea of the true parameter, so the most interesting part is the likelihood incorporating observed measurements. We assume that our measurements each have an error with Gaussian distribution and variance of 3cm around the true height at the measurement points. We further assume the measurement errors to be independent. As a result, for a specific model prediction, we expect the following distribution of possible measurements:

$$\mathcal{L}(\cdot|\theta) := \mathcal{N}\left(F(\theta), 0.03 \cdot I\right)$$

Combining prior and likelihood to form a posterior distribution, while notably leaving out the unknown scaling factor $\mathcal{P}(y)$ as discussed before, we now apply algorithm 1 to generate samples from that posterior distribution. This requires the choice of a proposal distribution, and we go with a mildly educated guess of

$$q(\cdot|\theta') \coloneqq \mathcal{N}(\theta', I).$$



(a) Samples from posterior distribution.

(b) Model states computed for the above set of parameters. Hits are indicated by green, misses by red markers.

Fig. 2.10: Markov Chain Monte Carlo posterior sampling and resulting model states.

Figure 2.10a shows the distribution of posterior samples obtained. From the shape of that distribution we can clearly see that velocity and angle have a rough inverse proportionality. This is reasonable since a faster throw at a lower angle could lead to similar measurements as a steeper but weaker throw. High velocity and high angle combined however would lead to an overshoot and therefore be very unlikely to produce the observed measurements.

So, the distribution of samples lets us estimate the parameters that would most likely explain our measurements in the Bayesian sense. This alone does not yet answer the question: How likely is the ball going to hit the basket given our measurements?

In order to answer this, we need to turn to our QOI again. Fortunately, now that we have a parameter distribution and are interested in values derived from model predictions, we are now in the same setting as forward UQ problems again (section 2.3.1). We already introduced MC methods in section 2.4.2, where we evaluate the model for a finite number of samples from the parameter distribution. So, we could now proceed to feed those samples into a dedicated MC method again. However, since we already have forward model evaluations for each of these samples, this work is already done! All that remains to do is to apply the QOI map to the samples, which typically means some post processing on the model states $F(\theta_i)$ we

already computed. In our example, this means checking for each trajectory whether it hits the basket, and computing the resulting expected value by taking the mean. The results for this particular experiment are shown in fig. 2.11.

Hits	Samples	Hit ratio
2803	10000	0.280

Fig. 2.11: Expected hit ratio determined from QOI via $\mathbb{E}[Q]$.

Just like before, computing more samples will deliver higher accuracy.

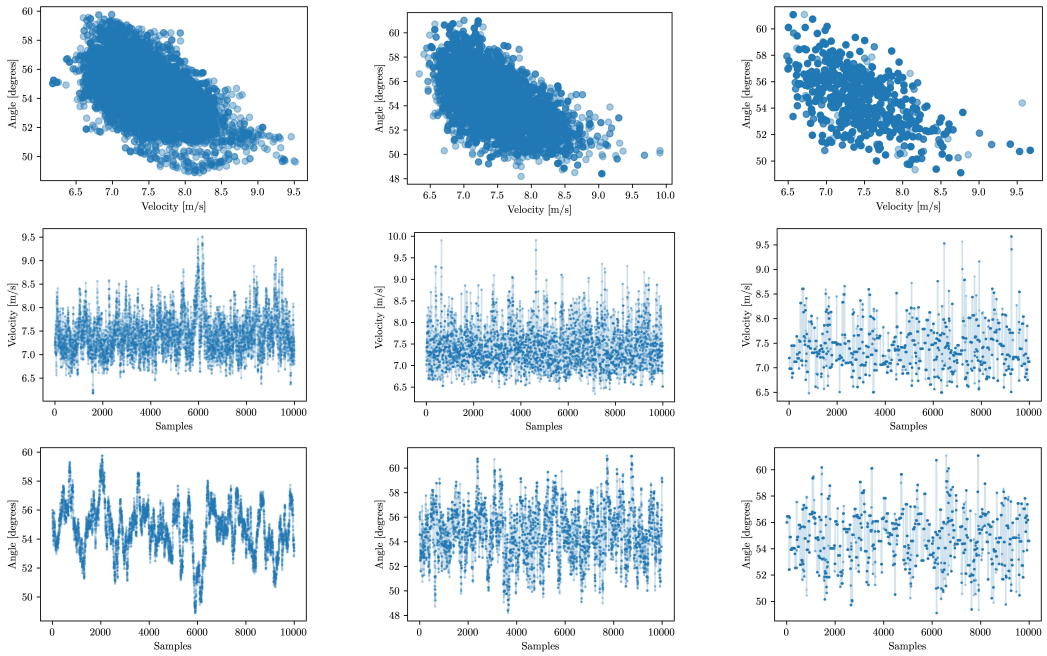
Efficiency considerations

We now have a working method, but how efficient is it? More specifically, we clearly produce correlated samples in the MCMC method. Closely correlated samples however add only very little information to the approximation we compute, whereas ideally we would like to generate fully independent samples. So, how close are we to drawing independent samples?

This can be estimated through the integrated autocorrelation time τ introduced in section 2.2, as well as the corresponding effective sample size. The integrated autocorrelation time estimates how many steps in the chain need to be taken on average in order to obtain a near-independent sample. The cost for obtaining N near-independent samples then is essentially τN forward model evaluations.

The MHMCMC method we used offers one main degree of freedom: The choice of the proposal distribution. It is intuitively clear that the proposal distribution affects the method's performance: If we choose a proposal distribution with an extremely small variance, subsequent samples will be extremely close to previous ones and have a high integrated autocorrelation time. As a result, the chain will remain in the same region of the parameter space for many steps, and we can only hope to cover the "weight" of the target distribution by computing a very large number of steps. Since each step requires a model evaluation, this comes at a very high computational cost. In the worst case, we might have to stop the chain while only having reached a small portion of the parameter space.

Conversely, a very high-variance proposal distribution may allow us to quickly jump across the parameter space. This comes at a high risk of proposing low-probability parameters far away from the high-probability regions. Many of those samples will not be accepted, and as a result we spend a large number of model evaluations on



ESS vel.	ESS ang.
147.6	39.7

(a) Low-variance proposal $q(\cdot|\theta') \coloneqq \mathcal{N}(\theta', 0.05 \cdot I)$.

ESS vel.	ESS ang.
483.7	154.3

(b) Mid-variance proposal $q(\cdot|\theta') \coloneqq \mathcal{N}(\theta', I)$.

ESS vel.	ESS ang.
298.6	167.5

(c) High-variance proposal $q(\cdot|\theta') \coloneqq \mathcal{N}(\theta', 20 \cdot I)$.

Fig. 2.12: Resulting samples (top), mixing plots showing the chains' behavior over time (mid) and effective sample sizes per parameter (bottom) for different proposal distribution variances.

samples that do not contribute to our approximate solution, again leading to a large integrated autocorrelation time.

Figure 2.12 illustrates this: Low-variance proposals lead to highly correlated proposals, many of which are accepted since they remain in approximately equally probable regions. High-variance proposals in turn are better at generating quasi-independent samples if accepted, but lead to lower acceptance rates, and as a result the chain tends to stagnate for multiple steps. We can further see that proposals behave differently regarding parameters: While the angle component of our parameter is traversed effectively by higher variance proposals, they become harmful to sample quality regarding initial velocity. This is easily explained by different magnitudes of values: After all, the likely velocity values span a smaller range of numbers than the likely angles.

Clearly, a more efficient and ideally problem independent type of proposal is desirable. While more involved regarding algorithm and implementation, the more

advanced approaches in chapter 3 all build upon the MHMCMC method presented here.

Convergence analysis

The convergence analysis of MCMC is well-established. Therefore, only a brief review is given here based on [Dod+19b, Section 2.1]. For a more in-depth treatment, refer to [RC10].

First, denote the transition kernel of the Markov chain $\{\theta^n\}_{n\in\mathbb{N}}$ as generated from algorithm 1 by

$$K(\theta'|\theta) \coloneqq \alpha(\theta'|\theta)q(\theta'|\theta) + \left(1 - \int_{\mathbb{R}^m} \alpha(\theta''|\theta)q(\theta''|\theta)d\theta''\right)\delta(\theta - \theta'),$$

where δ is the Dirac function. Further, let

$$\mathcal{E} = \{\theta : \pi_0(\theta) > 0\}, \mathcal{D} = \{\theta : q(\theta|\theta^*) > 0 \text{ for some } \theta^* \in \mathcal{E}\}.$$

Now $\mathcal{E}$ is the subset of $\mathbb{R}^m$ that the MCMC algorithm should sample from as defined by the prior. The set $\mathcal{D}$ in turn contains all parameter vectors that can possibly be reached via proposals. So, in order to ensure that the proposal distribution does not 'cut off' parts of the parameter space that are admitted in the prior, we assume $\mathcal{E} \subset \mathcal{D}$.

Lemma 2. *If $\mathcal{E} \subset \mathcal{D}$, then the posterior distribution π is a stationary distribution of the chain $\{\theta^n\}_{n\in\mathbb{N}}$.*

Proof. See [RC10]. □

With the condition $\mathcal{E} \subset \mathcal{D}$, it is easy to verify that by construction the transition kernel $K(\cdot|\cdot)$ satisfies the detailed balance condition

$$K(\theta|\theta^*)\pi_0(\theta^*) = K(\theta^*|\theta)\pi_0(\theta).$$

Theorem 3. *Assume $\mathbb{E}[|Q|] < \infty$ and*

$$q(\theta|\theta^*) > 0 \qquad \forall(\theta, \theta^*) \in \mathcal{E} \times \mathcal{E}. \tag{2.17}$$

Then it holds

$$\lim_{N\to\infty} \widehat{Q}_N^{MCMC} = \lim_{N\to\infty} \frac{1}{N+1} \sum_{i=0}^{N} \theta^i = \mathbb{E}_\nu[Q]$$

for any $\theta^0 \in \mathcal{E}$.

Proof. See [RC10]. □

These two results guarantee convergence of of the MCMC estimator to the expected value of the QOI. Note that, while this asymptotic result is entirely independent of the starting point $\theta^0 \in \mathcal{E}$, in practice a burnin phase is often desirable. That means choosing the estimator as

$$\widehat{Q}_N^{MCMC} = \lim_{N\to\infty} \frac{1}{N+1-n_0} \sum_{i=n_0}^{N} \theta^i,$$

and thereby ignoring the first n_0 samples generated by the MCMC algorithm. This can yield significant improvements in applications where the starting point θ^0 is chosen in a region where the posterior is relatively low. Due to correlation between samples, this can lead to that region being severely over-represented unless a sufficiently (and possibly prohibitive) number of samples is drawn. Choosing a sufficiently large n_0 instead allows the chain to "travel" towards a higher probability region before samples are taken into account.

The cost of the MCMC method is typically analyzed in relation to the MSE achieved, which is defined as

$$e(\widehat{Q}_N^{MCMC})^2 \coloneqq \mathbb{E}_\Theta[(\widehat{Q}_N^{MCMC} - \mathbb{E}_\rho[Q])^2].$$

Here, $\mathbb{E}_\Theta$ is the expected value with respect to the joint distribution of $\Theta \coloneqq \{\theta^n\}_{n\in\mathbb{N}}$ as generated by MCMC in algorithm 1.

We denote the computational ϵ-cost of the estimator by $\mathcal{C}(\widehat{Q}_N^{MCMC})^2 < \epsilon^2$, where the cost is understood in terms of floating point operations.

For further analysis, the MSE can be decomposed into

$$e(\widehat{Q}_N^{MCMC})^2 = \mathbb{V}_\Theta[\widehat{Q}_N^{MCMC}] + (\mathbb{E}_\Theta[\widehat{Q}_N^{MCMC}] - \mathbb{E}_\rho[Q])^2,$$

namely the variance of the MCMC estimator and the estimator's bias. In order to take into account the situation where the QOI can only be approximated, we can further exploit triangle inequality and linearity of expectation to arrive at

$$\begin{aligned} e(\widehat{Q}_N^{MCMC})^2 \leq \mathbb{V}_\Theta[\widehat{Q}_N^{MCMC}] + 2(\mathbb{E}_\Theta[\widehat{Q}_N^{MCMC}] - \mathbb{E}_{\nu^{M,R}}[\widehat{Q}_N^{MCMC}])^2 \\ + 2(\mathbb{E}_{\nu^{M,R}}[\widehat{Q}_N^{MCMC}] - \mathbb{E}_\rho[Q])^2, \end{aligned} \tag{2.18}$$

These terms now represent the three sources of error in the MCMC estimator. In particular, the last term is due to approximating the QOI Q by $Q_{M,R}$ and ρ by $\nu^{M,R}$. This is frequently the case in the Bayesian setting when the forward model cannot be solved exactly but instead only an approximation is accessible, for example in terms of a FE method. In this notation, M indicates the accuracy of the numerical approximation of the QOI, and R the dimension of the uncertain parameter. In the illustrative model from section 2.4.1, we have $R = 2$ due to the two-dimensional model parameter (angle and initial velocity) and $M = \frac{1}{h}$, where h is the step size of the explicit Euler method employed. Both M and R will become crucial in the analysis of multilevel methods in chapter 3, where they may vary across the hierarchy of models.

Variance and bias of the estimator (the first two terms) originate from the finite number of samples available and the fact that they are not i.i.d. samples from the target distribution $\nu^{M,R}$ respectively.

The sampling errors can be bounded asymptotically in the following way. Let $\widetilde{\theta}^0 \sim \nu^{M,R}$. Then the chain $\widetilde{\Theta} := \{\widetilde{\theta}^n\}_{n\in\mathbb{N}}$ generated via algorithm 1 from the starting point $\widetilde{\theta}^0$ is stationary, meaning $\widetilde{\theta}^n \sim \nu^{M,R}$ for $n \in \mathbb{N}$.

The latter implies that, for $\widetilde{Q}_{M,R}^n := Q_{M,R}(\widetilde{\theta}^n)$ we also have $\mathbb{V}_{\widetilde{\Theta}}[\widetilde{Q}_{M,R}^n] = \mathbb{V}_{\nu^{M,R}}[\widetilde{Q}_{M,R}]$ and $\mathbb{E}_{\widetilde{\Theta}}[\widetilde{Q}_{M,R}^n] = \mathbb{E}_{\nu^{M,R}}[\widetilde{Q}_{M,R}]$. Consequently, it holds

$$Cov_{\widetilde{\Theta}}[\widetilde{Q}_{M,R}^0, \widetilde{Q}_{M,R}^n] = \mathbb{E}_{\widetilde{\Theta}}[(\widetilde{Q}_{M,R}^0 - \mathbb{E}_{\nu^{M,R}}[\widetilde{Q}_{M,R}])(\widetilde{Q}_{M,R}^n - \mathbb{E}_{\nu^{M,R}}[\widetilde{Q}_{M,R})].$$

The asymptotic variance of the MCMC estimator is now defined as

$$\sigma_Q^2 := \mathbb{V}_{\nu^{M,R}}[\widetilde{Q}_{M,R}] + 2\sum_{n=1}^{\infty} Cov_{\widetilde{\Theta}}[\widetilde{Q}_{M,R}^0, \widetilde{Q}_{M,R}^n].$$

Note that in the idealized case of i.i.d. samples it holds $\sigma_Q^2 = \mathbb{V}_{\nu^{M,R}}[\widetilde{Q}_{M,R}]$, because samples are not correlated in that case. Further, observe that the asymptotic variance relates to the integrated autocorrelation time τ_Q of the chain (see eq. (2.16)) by

$$\frac{\sigma_Q^2}{\mathbb{V}_{\nu^{M,R}}[\widetilde{Q}_{M,R}]} = \tau_Q$$

Theorem 4 (central limit theorem)**.** *Assuming eq.* (2.17) *and* $\sigma_Q^2 < \infty$ *and*

$$\mathbb{P}[\alpha^{M,R} = 1] < 1$$

.

Then it holds

$$\sqrt{N}(\widehat{Q}_N^{MCMC} - \mathbb{E}_{\nu^{M,R}}[Q_{M,R}]) \xrightarrow{D} \mathcal{N}(0, \sigma_Q^2)$$

Proof. See [RC10]. □

This result implies that the sampling errors of the MCMC estimator asymptotically decays at the same rate as an idealized estimator using i.i.d. samples. We are therefore (at least in an asymptotic sense) in the same situation as MC again, except for requiring an additional constant factor in the number of samples due to the MCMC samples' correlation.

In order to finally arrive at a cost bound of the MCMC involving a finite number of samples, we now have to leave the asymptotic regime and instead make the assumption

Assumption 1. *For any* $N \in \mathbb{N}$*, we can bound the sampling error by*

$$\mathbb{V}_\Theta[\widehat{Q}_N^{MCMC}] + 2(\mathbb{E}_\Theta[\widehat{Q}_N^{MCMC}] - \mathbb{E}_{\nu^{M,R}}[\widehat{Q}_N^{MCMC}])^2 \lesssim \frac{\mathbb{V}_{\nu^{M,R}}[Q_{M,R}]}{N}.$$

The hidden constant is assumed to be independent of M*,* R *and* N*.*

Such error bounds are not trivial to derive, but have been shown for certain settings (e.g. [Rud12]). Note that, while the constant in assumption 1 is assumed independent of M, R and N, it will in many cases directly depend on the autocorrelation τ_Q (or, equivalently, σ_Q^2) due to the aforementioned correlation between MCMC samples.

Now the final term in eq. (2.18) remaining to be controlled is the discretization bias.

Assumption 2. *We assume* $\mathbb{E}_{\nu^{M,R}}[Q_{M,R}] \to \mathbb{E}_\rho[Q]$ *for* $M, R \to \infty$ *with convergence order* $\alpha, \alpha' > 0$ *in the sense of*

$$|\mathbb{E}_{\nu^{M,R}}[Q_{M,R}] - \mathbb{E}_\rho[Q]| \lesssim M^{-\alpha} + R^{-\alpha'}$$

Under assumptions 1 and 2, it follows for the MSE from eq. (2.18)

$$e(\widehat{Q}_N^{MCMC}) \lesssim \frac{\mathbb{V}_{\nu^{M,R}}[Q_{M,R}]}{N} + M^{-\alpha}$$

Adding the mild assumption that $\mathbb{V}_{\nu^{M,R}}$ is bounded independent of M and R, it now suffices to choose $N \approx \epsilon^{-2}$ and $M \approx \epsilon^{-\frac{1}{\alpha}}$ to conclude $e(\widehat{Q}_N^{MCMC}) = \mathcal{O}(\epsilon^2)$.

In a final step, a bound on the computational cost needed to achieve a given error tolerance can be derived. Assume that computing one sample of $Q_{M,R}^n$ has a cost of $\mathcal{C}(Q_{M,R}^n) \leq M^\gamma$ for a $\gamma > 0$. Again choosing $N \approx \epsilon^{-2}$ and $M \approx \epsilon^{-\frac{1}{\alpha}}$, we can conclude that the cost associated with an error tolerance

$$e(\widehat{Q}_N^{MCMC})^2 < \epsilon^2$$

can be bounded by

$$Cost(\widehat{Q}_N^{MCMC}) \lesssim NM^\gamma \approx \epsilon^{-2-\gamma/\alpha}. \tag{2.19}$$

As before, the hidden constant typically depends on the integrated autocorrelation time τ_Q of the chain.

When applied to the model from section 2.4.1, the same constants and resulting growth in cost may be expected as in the MC case from section 2.4.2, i.e. $\alpha = \gamma = 1$. Again, this directly follows from the convergence rate and cost scaling of the explicit Euler method employed, and we arrive at a rate $\mathcal{O}(\epsilon^{-3})$.

From this analysis, it is clear that the overall cost of the MCMC method for a specific problem is determined by

- the cost of individual samples, which may grow significantly when accurate numerical approximations of the forward model are needed in order to control the discretization bias,

- and the number of samples needed to control the sampling error, driven mainly by the chain's integrated autocorrelation time τ_Q.

More advanced approaches attempting to mitigate these sources of cost are discussed in chapter 3. In fact, they mainly adress the two issues above: Hierarchies of forward models are used in order to shift most of the sampling work to cheaper models, and cheaper models are used in order to generate good proposals, reducing autocorrelation time.

3 Advanced Uncertainty Quantification Methods

The basic Monte Carlo (MC) and Markov Chain Monte Carlo (MCMC) algorithms from section 2.4 already serve the purpose of solving both forward and inverse Uncertainty Quantification (UQ) problems respectively, and provably so under very mild conditions.

As already discussed in section 2.4.2 and section 2.4.3, these methods however exhibit bad scaling of cost per accuracy when applied to computationally expensive forward models. For intricate parameter distributions and, most importantly, higher dimensional parameters, this becomes even more pronounced in practice. Working towards the main goal of this thesis, namely advancing UQ on forward models that are computationally challenging themselves, we therefore need to turn towards more efficient methods.

In this chapter we begin with an overview and comparison of existing UQ methods, and choose Multilevel Markov Chain Monte Carlo (MLMCMC) methods as the ones best fulfilling our goals. We proceed with a review of existing work in that field and conclude with the proposal of an improved Multiindex Markov Chain Monte Carlo (MIMCMC) method.

For the treatment of large-scale problems, both the stochastic side and the forward model need to be considered. Here we focus mainly on the stochastic side, while chapter 4 aims to complement our multilevel and multiindex UQ algorithms with carefully designed hierarchical forward models built specifically for that purpose.

Parts of this chapter have been published by the author in [See+21].

3.1 Methods Overview

For both forward and inverse UQ problems, a wide range of UQ methods have been developed, and rightfully so: There is no single best approach, since they all vary extremely in how they balance the following three contradictive aspects.

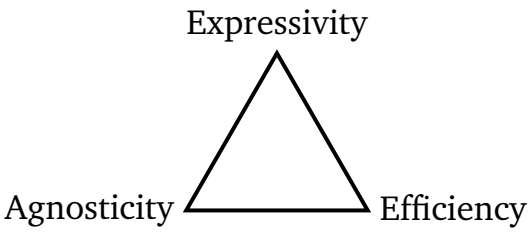


Fig. 3.1: The three main aspects of UQ methods.

- Expressivity: In both forward and inverse UQ problems we are interested in obtaining a probability distribution. Some methods deliver approximation to the entire distribution, for example in terms of samples or as linear combinations of simple distributions like Gaussians. Others, however, may only deliver certain statistical moments of the distribution or the Maximum a posteriori probability (MAP) point. The appropriate choice here depends strongly on the particular application.
- Efficiency: UQ methods themselves tend to be of relatively low computational cost, while forward models can incur substantial computational effort. Therefore, the main difference lies in how many model evaluations are required. For example, optimization based methods have a clear advantage over sampling based methods, since they do not need to recover the entire distribution and may directly iterate towards a peak.
- Agnosticity: Some methods may only require simple evaluations of the forward model F and are model agnostic in the sense that minimal assumptions on the forward model are made. Others may additionally require derivatives, or even a fundamental reformulation of the forward problem. In practice, the latter may not be feasible with more complex models, while the former methods may lend themselves to coupling with existing model implementations.

In the following we give an overview on various UQ methods and classify them according to these three properties. Regarding forward problems, MC methods require only forward model evaluations to recover a full pushforward distribution, but can be extremely computationally expensive, as we already saw in section 2.4.2.

On the other hand, Stochastic Galerkin type methods [GS91; XK03] allow efficiently recovering the entire distribution and, in suitable settings, can be very efficient. They do however require a significant modification to the model itself, since they essentially treat the uncertain parameters' space as part of the Partial Differential Equation (PDE) domain.

Another approach, stochastic collocation, can work with simple model evaluations and recover the target distribution (for inverse problems as well [MX09]), but is efficient only for lower dimensional parameter spaces [Xiu15].

For inverse problems, there are highly efficient optimization-based methods for determining the MAP point in inverse problems using derivatives of the posterior [Kle+17; Kit95; LK14]. They are extremely efficient, but derivatives of the posterior and thereby of the inverse model map can be hard to obtain.

Stochastic collocation type methods can work with simple model evaluations and recover the posterior [MX09], but become inefficient for higher dimensional parameter spaces [Xiu15].

MCMC type methods in turn recover the full posterior distribution and in many cases only need simple forward model evaluations, but come at a very high computational cost in terms of numerous model evaluations required to achieve a good approximation.

For the rest of this thesis, we choose MC and MCMC class methods for their agnosticity and expressivity. Clearly, this incurs high computational demand. We seek to mitigate this in the following, with a particular focus on inverse problems and MCMC.

There is one main approach to reduce the number of model evaluations in MCMC: Metropolis-Hastings Markov Chain Monte Carlo (MHMCMC) produces correlated samples, since each proposed step is in the vicinity of the previous one. Strongly correlated samples barely contribute information to our approximation (in the most extreme case of identical samples, we gain nothing over a simple sample). So, as already indicated in a very simple case in section 2.4.3, less correlated proposals and high acceptance rates would allow us to achieve a good approximation with fewer samples. A wide variety of improved proposals aims to achieve this. Examples include preconditioned Crank-Nicolson [Bes+08; Cot+13; RS15], Adaptive Metropolis [HS98; HST01], Hamiltonian MCMC [Dua+87], Dimension-Independent Likelihood-Informed (DILI) MCMC [CLM16; CDS19], and many others. They largely come down to forming some kind of understanding of the shape of the posterior, and using it to generate more well-informed proposals.

Rather than minimizing the number of model evaluations for a single model, Multilevel Monte Carlo (MLMC) [Hei01; Gil08] and MLMCMC [Dod+15; Dod+19b] take a second and somewhat orthogonal route: An entire hierarchy of models is defined, ranging from cheap to compute rough approximations to the accurate, yet expensive, full model. MLMC and MLMCMC make no assumptions on what exactly those coarse

models could be. For example, we could use suitable Ordinary Differential Equation (ODE)s as rough approximations to a more complex PDE model, as long as model predictions are sufficiently close. We will clarify the specific requirements in the following sections and present some particularly efficient choices of model hierarchies in chapter 4. A more obvious choice for a level hierarchy could be mesh width in numerical PDE solvers, where theoretical a priori error bounds typically guarantee that coarser meshes still deliver reasonable approximations, and for certain cases convergence results can be shown. For an example of such a theoretical derivation, we refer to the Poisson equation example investigated in [Dod+15; Dod+19b].

3.2 Multilevel Monte Carlo

As a prelude to presenting the MLMCMC algorithm and theory, we begin with a quick introduction of MLMC [Hei01; Gil08], its corresponding and somewhat simpler algorithm for forward UQ. Both have in common that, instead of working on a single forward model F, they introduce an entire hierarchy of models

$$F_0, F_1, \ldots \qquad \text{with} \qquad F_l : \mathbb{R}^{m_l} \to \mathbb{R}^{n_l} \qquad \text{and} \qquad \lim_{l\to\infty} F_l = F.$$

Each of these model maps obviously implies its own model prediction $F_i(\theta)$ approximating $F(\theta)$, whose distribution now depends on the prior distribution and the respective model F_i.

We refer to model maps F_l with low and high index l as coarser models and finer models respectively. For the convergence results of the following multilevel algorithms it is crucial that, for increasing l, the models' approximation error decreases exponentially while the computational cost increases exponentially. In practice, these requirements may be relaxed to some degree.

Further, in order to seamlessly apply asymptotical results from e.g. PDE theory, we now have to differentiate between the theoretical "exact" model F and the finest numerical approximation F_L we can afford to compute numerically.

This model hierarchy can be exploited by replacing the expected value of our finest model outcome by the following telescoping sum:

$$\mathbb{E}[F(\theta)] \approx \mathbb{E}[F_L(\theta)] = \underbrace{\mathbb{E}[F_0(\theta)]}_{\text{Coarse approximation}} + \sum_{l=1}^{L} \underbrace{\mathbb{E}[F_l(\theta) - F_{l-1}(\theta)]}_{\text{Fine corrections}}$$

Clearly this telescoping sum is an identity, so what have we gained? Here our previous assumption on the model hierarchy come into play: The coarsest component of the telescoping sum requires many samples to be drawn, since it has to recover the entire distribution. However, since the coarsest model is cheap to compute, overall cost remains small. With the subsequent components, computational effort increases. But here we can exploit that we are essentially sampling from the difference between two approximations: Since we assume finer models to approximate each other better and better, the increase in cost for a single evaluation is counteracted by less and less samples being required to recover the respective difference accurately.

In order to exploit this variance reduction across levels in practice, we need to use the same parameter sample to compute one sample of a difference between levels. So, we arrive at the following MLMC estimator:

$$\mathbb{E}[F(\theta)] \approx \widehat{F}_{N,L}^{MLMC} := \frac{1}{N_0}\sum_{i=1}^{N_0} F_0(\theta_0^i) + \sum_{l=1}^{L}\frac{1}{N_l}\sum_{i=1}^{N_l}(F_l(\theta_l^i) - F_{l-1}(\theta_l^i)),$$

where θ_l^i is the ith sample drawn for level l from our prior distribution. For this estimator, the following cost estimate is proven in [Gil08, Thm 3.1]. It is based on relating model error to model cost, denoting the discretization levels by an sequence M_l increasing with l.

Theorem 5 (MLMC cost)**.** *Let $\epsilon < e^{-1}$ and suppose there are constants $\alpha, \beta, \gamma > 0$ with $\alpha \geq \frac{1}{2}\min(\beta, \gamma)$ such that, for all $l \geq 0$:*

- $|\mathbb{E}[F_l - F]| = \mathcal{O}(M_l^{-\alpha})$,
- $\mathbb{V}[F_{l+1} - F_l] = \mathcal{O}(M_l^{-\beta})$, *and*
- $Cost(F_l(\theta^i)) = \mathcal{O}(M_l^{\gamma})$.

Then there exist $L \in \mathbb{N}$ and $\{N_l\}_{l=0}^{L} \subset \mathbb{N}$, such that for the mean square error it holds $\mathbb{E}[(\widehat{F}_{N_l,L}^{MLMC} - \mathbb{E}[F])^2] < \epsilon^2$ and the overall cost of the multilevel estimator is given by

$$Cost(\widehat{F}_{N_l,L}^{MLMC}) = \begin{Bmatrix} \mathcal{O}(\epsilon^{-2}) & for & \beta > \gamma, \\ \mathcal{O}(\epsilon^{-2} log(\epsilon)^2) & for & \beta = \gamma, \\ \mathcal{O}(\epsilon^{-2-\frac{\gamma-\beta}{\alpha}}) & for & \beta < \gamma. \end{Bmatrix}$$

So, for a suitably chosen model hierarchy, we can achieve a cost estimate significantly better than the MC counterpart $Cost(\widehat{F}_N^{MC}) \propto N = \mathcal{O}(\epsilon^{-2-\frac{\gamma}{\alpha}})$ from section 2.4.2. For example, for the ODE model in section 2.4.1, a model hierarchy with $M_l =$

$2^l \propto \frac{1}{h_l}$ can be constructed by simply cutting step sizes in half across levels, i.e. $h_l = \frac{1}{2}h_{l-1}$, starting from a coarse level step size h_0. In that situation, we have $\alpha = 1$, $\beta = 2$ and $\gamma = 1$ due to employing an explicit Euler method with linear global convergence and linear increase in cost with $\frac{1}{h}$. Due to $\beta > \gamma$, the overall cost of MLMC will scale with ϵ^{-2} in contrast to ϵ^{-3} in the single level MC case.

Note that the assumptions in theorem 5 are the only requirement on the relation between forward models of different levels. As already mentioned earlier, it is therefore not relevant how these models are actually constructed; what matters is that, in terms of mean and variance, they approximate each other in a way that is suitably balanced with the respective cost per model evaluation.

3.3 Multilevel Markov Chain Monte Carlo

Extending the MLMC algorithm above to an MCMC type method for inverse problems could be done in a trivial way: Since MLMC requires samples to be drawn on various levels, we could simply run MCMC chains on each of those levels. Here we still benefit from the fact that more but cheaper samples are needed on coarser levels, while more expensive but fewer samples are needed on finer levels.

As already introduced in section 2.3, the goal is to estimate a quantity of interest Q. The multilevel method works on a hierarchy of forward models F_i, and therefore we have an associated sequence of Quantity of Interest (QOI) maps

$$Q_0, Q_1, ... \qquad \text{with} \qquad \lim_{l \to \infty} Q_l = Q.$$

Note that particular care has to be taken regarding distributions. While in the forward setting forward models differ, but the parameter distribution to sample from is the same, here we have different posterior distributions across levels because the posterior's likelihood component depends on the respective forward model F_i.

$$\mathbb{E}[Q] \approx \mathbb{E}_{\nu_L}[Q_L] = \underbrace{\mathbb{E}_{\nu_0}[Q_0]}_{\text{Coarse approximation}} + \sum_{l=1}^{L} \underbrace{\mathbb{E}_{\nu_l}[Q_l] - \mathbb{E}_{\nu_{l-1}}[Q_{l-1}]}_{\text{Fine corrections}} \tag{3.1}$$

For subsequent analysis, we further specify the construction of levels by defining $Q_l \coloneqq Q_{M_l,R_l}$ where, as in section 2.4.3, M_l corresponds to the refinement of the numerical forward model approximation and R_l is the dimension of the model

parameter on level l. Note that, for readers new to multilevel UQ methods, it is enough to consider the case of constant parameter dimension across levels (i.e. $R_l = R$) for a basic understanding of the algorithm.

While M_l typically corresponds to $\frac{1}{h}$ (for mesh width h) in Finite element (FE) models, we will only require M_l to relate computational cost to approximation error in a suitable way. Further denote the posterior distribution on each level by $\nu^l := \nu^{M_l,R_l}$, and the corresponding densities by $\pi^l := \pi^{M_l,R_l}$.

There is, however, another approach to exploiting hierarchies in MCMC methods: Coarser levels can be used to produce well-informed proposals for finer levels. A simple method could be sampling from a coarse level, and only use a subset of those (i.e. subsampling) as proposals for a fine level. As a result, fine proposals will be nearly independent and we could achieve good mixing on the fine level with few expensive samples. One method in this spirit, even though more refined, is delayed acceptance MCMC [CF05].

Incorporating this is the key ingredient in MLMCMC as proposed in [Dod+15; Dod+19b]. This section follows the presentation in [Dod+19b, Section 3].

We begin by defining an MCMC type estimator for each term in the telescoping sum in eq. (3.1).

The coarsest component, $\mathbb{E}_{\nu_0}[Q_0]$, may be estimated by single level MCMC as already discussed in section 2.4.3. We denote the resulting estimator for level zero by $\widehat{Q}^{MCMC}_{0,N_0}$, where N_0 indicates the number of samples drawn.

For estimating the correction terms, define $Y_l := Q_l - Q_{l-1}$. The corresponding estimator is the chosen as

$$\widehat{Y}^{MCMC}{}_{l,N_l} := \frac{1}{N_l}\sum_{n=1}^{N_l} Y_l^n = \frac{1}{N_l}\sum_{n=1}^{N_l} Q_l(\theta_l^n) - Q_{l-1}(\Theta_{l-1}^n) \tag{3.2}$$

where N_l is the number of samples on level l, and we make use of two Markov chains per level: A chain $\{\theta_l^n\}_{n\in\mathbb{N}}$ on level l itself as well as an associated coarse chain $\{\Theta_{l-1}^n\}_{n\in\mathbb{N}}$, where each have the appropriate stationary distribution (ν_l and ν_{l-1} respectively).

Finally, by adding all estimators, we arrive at the MLMCMC estimator:

$$\widehat{Q}^{MLMCMC}_{L,\{N_l\}} := \widehat{Q}^{MCMC}_{0,N_0} + \sum_{l=1}^{L} \widehat{Y}^{MCMC}{}_{l,N_l}. \tag{3.3}$$

It remains to define how the respective Markov chains for the estimators $\widehat{Y}^{MCMC}{}_{l,N_l}$ are to be generated. To that end, we will proceed in two main steps. In a first step, focusing on coupling two levels, we define how the fine components are to be sampled, assuming idealized Independent and identically distributed (i.i.d.) coarse samples Θ_{l-1}^n serving as proposals.

Sampling the correction terms

For simplicity, consider a fixed level $1 \leq l \leq L$. In order to accommodate varying parameter dimensions across levels, i.e. R_l varies across levels, we partition the fine chain consisting of samples $\theta_l \in \mathbb{R}^{R_l}$ into fine and coarse components:

$$\theta_l = [\theta_{l,C}, \theta_{l,F}],$$

where the coarse component matches the dimension on the coarse level $l-1$, i.e. $\theta_{l,C} \in \mathbb{R}^{R_{l-1}}$. As a result, the fine component $\theta_{l,F}$ has dimension $R_l - R_{l-1}$.

Now we can exploit the coarse level in order to provide well-informed proposals for the fine level (at least for the coarse component $\theta_{l,C}$) as well as ensure variance reduction between levels as $l \to \infty$. Both advantages ultimately come down to the fact that ν_{l-1} is assumed to be a good approximation of ν_l.

Ideal proposals would be i.i.d. samples from the target distribution itself; in that case, every proposal could be accepted, trivially yielding i.i.d. samples from the target distribution. Drawing proposals from the next coarser posterior distribution approximates that idealized situation as $l \to \infty$ when posteriors can be assumed to "converge" in a sense specified later. For the same reason, many proposals will be accepted, meaning that coarse components on level l often equal their corresponding proposals from level $l-1$. Under assumptions specified later, that implies similar quantities of interest on both levels, in turn implying low variance.

The approach is detailed in algorithm 2, where for now we assume i.i.d. samples from ν_{l-1} are readily available and specify later how to obtain them in an approximate sense.

The algorithm comes down to drawing a sample Θ_{l-1}^{n+1} from the level $l-1$ posterior distribution and making it the proposal $\theta'_{l,C}$ for the coarse component on level l. In order to support the case $R_l > R_{l-1}$, the proposal's fine component '$\theta'_{l,F}$ needs to be specified as well. To that end, we draw $\theta'_{l,F}$ from a level-specific proposal

Algorithm 2: MHMCMC for Y_l

Result: Samples $\{\theta_l^n\}$ and $\{\Theta_{l-1}^n\}$ as needed by the estimator $\widehat{Y}^{MCMC}{}_{l,N_l}$.
Choose initial states $\Theta_{l-1}^0 \sim \nu^{l-1}$ and $\theta_l^0 := [\Theta_{l-1}^0, \theta_{l,F}^0]$.
for *sample* $n = 0, \dots, N_l$ **do**

- Assume an i.i.d. sample Θ_{l-1}^{n+1} from ν^{l-1}.
- Given θ_l^n, generate θ_l^{n+1} in analogy to algorithm 1. In that process, choose the proposal distribution $q_{ML}^l(\theta_l'|\theta_l^n)$ implicitly defined by taking $\theta_{l,C}' := \Theta_{l-1}^{n+1}$ and generating θ_l', F from a proposal distribution $q_{ML}^{l,F}(\theta_{l,F}', \theta_{l,F}^n)$ independent of the coarse component. As acceptance probability, use

$$\alpha_{ML}^l(\theta_l'|\theta_l^n) = min\left\{1, \frac{\pi^l(\theta_l')q_{ML}^l(\theta_l^n|\theta_l')}{\pi^l(\theta_l^n)q_{ML}^l(\theta_l'|\theta_l^n)}\right\}.$$

end

distribution $q_{ML}^{l,F}$. Combining coarse and fine component, we arrive at our full level l proposal.

In order to arrive at an explicit formulation of the proposal q_{ML}^l, we can break it down in the following way.

Lemma 3. *The acceptance probability α_{ML}^l from algorithm 2 can be written as*

$$\alpha_{ML}^l(\theta_l'|\theta_l^n) = min\left\{1, \frac{\pi^l(\theta_l')\pi^{l-1}(\theta_{l,C}^n)q_{ML}^{l,F}(\theta_{l,F}^n|\theta_{l,F}')}{\pi^l(\theta_l^n)\pi^{l-1}(\theta_{l,C}')q_{ML}^{l,F}(\theta_{l,F}'|\theta_{l,F}^n)}\right\}.$$

and the induced transition kernel K_{ML}^l satisfies detailed balance.

For symmetric fine level proposals this fraction simplifies to

$$\alpha_{ML}^l(\theta_l'|\theta_l^n) = min\left\{1, \frac{\pi^l(\theta_l')\pi^{l-1}(\theta_{l,C}^n)}{\pi^l(\theta_l^n)\pi^{l-1}(\theta_{l,C}')}\right\}.$$

Note that for symmetric proposals however, the convergence analysis in [Dod+19b] does not hold (see [Dod+19a]). In the applications, we therefore restrict our use of symmetric proposals to the coarsest level.

Proof. Proposals $\theta_{l,C}'$ for coarse components and proposals $\theta_{l,F}'$ for fine components are, by construction, independent. Therefore the overall proposal density q_{ML}^l can

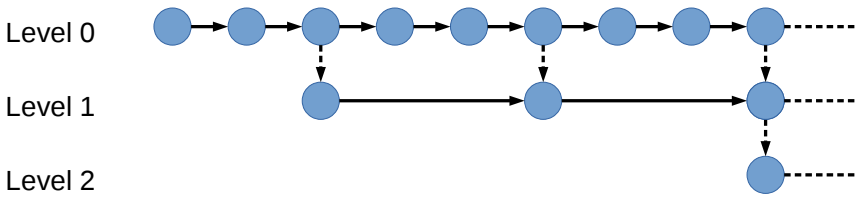


Fig. 3.2: Structure of samples being passed across levels of MLMCMC.

be decomposed into a product of a coarse proposal density $q_{ML}^{l,C}$ and a fine proposal density $q_{ML}^{l,F}$.

While the latter may be chosen in a variety of ways, coarse proposals in algorithm 2 are defined as being true i.i.d. samples from the coarse posterior π^{l-1}. Consequently, it follows that $q_{ML}^{l,C}(\theta'_{l,C}|\theta^n_{l,C}) = \pi^{l-1}(\theta'_{l,C})$ and $q_{ML}^{l,C}(\theta^n_{l,C}|\theta'_{l,C}) = \pi^{l-1}(\theta^n_{l,C})$. The kernel K_{ML}^l fulfilling the detailed balance condition then follows in analogy to MHMCMC (see section 2.4.3).

□

Recursive generation of coarse proposals

Algorithm 2 fulfills the purpose of coupling samples from a coarser level $l-1$ to a level l chain, assuming i.i.d. samples from the coarse posterior are available. This is a helpful step in constructing a valid MCMC method for sampling the correction term estimator. However, these samples are clearly not available in practice, and will have to be approximated.

An obvious practical way to obtain them is simply applying standard MHMCMC to the coarser level, leading to a two-level method estimating each correction term. Instead of plain MHMCMC, we can further apply algorithm 2 on level $l-1$ in order to produce proposals needed on level l. By applying this approach recursively, we finally arrive at a full multilevel method in algorithm 3. How samples are passed across levels is illustrated in fig. 3.2.

Since MCMC methods produce correlated samples, the i.i.d. assumption is however violated. In order to mitigate this, subsampling is applied; i.e. instead of using all samples $\{\Theta^n_{l-1}\}_{n\in\mathbb{N}}$ of the coarse chain directly, we instead use the subset $\{\Theta^{\rho_{l-1}n}_{l-1}\}_{n\in\mathbb{N}}$ for a subsampling rate $\rho_{l-1} \in \mathbb{N}$. Now, if $\rho_{l-1} \gg \tau_{l-1}$ where τ_{l-1} is the

Algorithm 3: Multilevel MCMC

Result: Markov chains $\{\theta_k^i\}_{i=0}^{N_k}$ for all levels $k \in \{0, ..., L\}$.
On level 0, run a conventional MCMC chain, delivering samples θ_0^i.
for *level* $k = 1, \ldots, L$ **do**
  Choose starting point θ_k^0 with coarse component from next coarser starting point θ_{k-1}^0.
  for *sample* $j = 1, \ldots, N_k$ **do**
    Given θ_k^j, generate proposal $\theta_k' = \left\{ \begin{array}{c} \theta_{k,C}' \\ \theta_{k,F} \end{array} \right\}$ where

    - $\theta_{k,C}'$ is drawn from chain on level $k-1$, observing subsampling, and
    - $\theta_{k,F}'$ from proposal density $q_k(\theta_{k,F}'|\theta_{k,F}^j)$.

    Compute acceptance probability

$$\alpha(\theta_k', \theta_k^j) = \min \left\{ 1, \frac{\pi_k(\theta_k') q_k(\theta_k^j|\theta_k')}{\pi_k(\theta_k^j) q_k(\theta_k'|\theta_k^j)} \cdot \frac{\pi_{k-1}(\theta_{k,C}^j)}{\pi_{k-1}(\theta_{k,C}')} \right\}.$$

    Draw a random number $r \in [0, 1]$.
    if $r < \alpha(\theta_k', \theta_k^j)$ **then**
      Accept proposal: θ_k' as θ_k^{j+1}
    else
      Reject proposal: $\theta_k^{j+1} = \theta_k^j$
    end
  end
end

integrated autocorrelation time of the level $l-1$ chain, we obtain approximately uncorrelated coarse samples. In practical applications, a value around $\rho_{l-1} \approx 1.5\tau_{l-1}$ often turns out to be a good choice.

For theoretical analysis of the algorithm, i.i.d. coarse samples will still be assumed in the following. The bias introduced by drawing slightly correlated samples from an MCMC chain on the coarse level is therefore not covered, but turns out negligible in practical applications.

Convergence analysis

The convergence analysis from [Dod+19b] is conducted for the idealized algorithm 2. That is, i.i.d. samples from level $l-1$ are assumed when generating level l samples. As already mentioned, this assumption is clearly not realistic, since generating samples for any level is the goal in the first place. Algorithm 3 can be used in practice, approximating i.i.d. proposals by subsampling the coarse chain. This approximation naturally incurs an error. It is, however, beyond the scope of this work to address that issue. We therefore continue with the analysis for the idealized method, tacitly assuming in practical applications that appropriate subsampling rates render the effect of this discrepancy negligible.

Convergence analysis of MLMCMC according to [Dod+19b] can be conducted in the same spirit as MCMC in section 2.4.3. For each level $l \in 1, \dots, L$, let

$$\begin{aligned}\mathcal{E}^l &= \{\theta_l : {\pi_0}^l(\theta_l) > 0\},\\ \mathcal{D}^l &= \{\theta_l : q^l_{ML}(\theta_l|\theta^*_l) > 0 \text{ for some } \theta^*_l \in \mathcal{E}^l\}.\end{aligned}$$

Then the following results follow from their single level MCMC counterparts (see section 2.4.3) applied to each chain. Applied to the telescoping sum structure in eq. (3.1) this implies the desired properties.

Lemma 4. *Assuming $\mathcal{E}^l \subset \mathcal{D}^l$, ν^l is a stationary marginal distribution of the Markov chain $\{\theta^n_l\}_{n\in\mathbb{N}}$.*

Theorem 6. *Assume for $l \in \{1, \dots, L\}$ it holds $\mathbb{E}_{\nu^l}[|Q_l|] < \infty$ and*

$$q^l_{ML}(\theta_l|\theta^*_l) > 0 \qquad \forall(\theta_l, \theta^*_l) \in \mathcal{E}^l \times \mathcal{E}^l. \tag{3.4}$$

Then it holds

$$\lim_{\{N_l\}\to\infty} \widehat{Q}_{L,\{N_l\}}^{MLMCMC} = \mathbb{E}_{\nu^L}[Q_L]$$

for any $\theta_l^0 \in \mathcal{E}^l$.

The assumption in eq. (3.4) is somewhat more intricate than its single level counterpart in eq. (2.17). As already seen above, it holds

$$q_{ML}^l(\theta_l|\theta_l^*) = \pi^{l-1}(\theta_{l,C}) q_{ML}^{l,F}(\theta_{l,F}|\theta_{l,F}^*). \tag{3.5}$$

Consequently, the assumption actually couples two levels in the sense that on the level l posterior's support $\mathcal{E}^l$, both the fine proposal and, most importantly, the coarse posterior must be positive as well.

At first glance, eq. (3.4) is just as mild an assumption for practical applications as the single level case eq. (2.17). For example, choosing Gaussian distributions for prior, likelihood and fine proposals on all levels ensures that the respective densities are all positive on the entire parameter space.

However, there is a severe implication when only a finite number of samples may be drawn. Assume that $\pi^l \gg 0$ and $\pi^{l-1} \approx 0$ holds in a significant region $\mathcal{D}_{low}^l$ of the parameter space. We also have $q_{ML}^l \approx 0$ in $\mathcal{D}_{low}^l$ due to eq. (3.5). This essentially violates the assumption $\mathcal{E}^l \subset \mathcal{D}^l$ made in lemma 4 in the finite setting. Consequently, the posterior ν^l will not be represented well by sampling unless drawing potentially prohibitively large numbers of samples. Therefore, in applications like the ones in chapter 6 it must be ensured that the coarse posterior "covers" the weight of the fine posterior well. This may be challenging depending on the choice of forward models and likelihoods. A more detailed investigation of this issue as well as an approach to (at least partially) compensate for it can be found in [Lyk+20].

Finally, the following cost estimate for the idealized method is proven in [Dod+19b]. It requires the following objects: Let $\Theta_l := \{\theta_l^n\}_{n\in\mathbb{N}} \cup \{\Theta_{l-1}^n\}_{n\in\mathbb{N}}$ for $l \geq 1$, and $\Theta_0 := \{\theta_0^n\}_{n\in\mathbb{N}}$. Denote the expected value and variance with respect to Θ_l (generated by algorithm 2) by $\mathbb{E}_{\Theta_l}$ and $\mathbb{V}_{\Theta_l}$. Further, denote by $\nu^{l,l-1}$ the joint distribution of θ_l and Θ_{l-1}, which is defined by the marginals of θ_l and Θ_{l-1} (namely ν^l and ν^{l-1}) with a correlation implied by algorithm 2. For clarity of presentation, further let $Y_0 := Q_0$, $\nu^{0,-1} := \nu^0$, and $M_{-1} = R_{-1} = 1$.

Theorem 7 (MLMCMC cost)**.** *Let $\epsilon < e^{-1}$ and suppose there are constants $\alpha, \alpha', \beta, \beta', \gamma > 0$ with $\alpha \geq \frac{1}{2}\min(\beta, \gamma)$ such that, for all $l \geq 0$:*

- $|\mathbb{E}_{\nu^l}[Q_l] - \mathbb{E}_{\nu}[Q]| \leq C(M_l^{-\alpha} + R_l^{-\alpha'})$,
- $\mathbb{V}_{\nu^{l,l-1}}[Y_l] \leq C(M_{l-1}^{-\beta} + R_{l-1}^{-\beta'})$,
- $\mathbb{V}_{\Theta_l}[\widehat{Y}^{MCMC}{}_{l,N_l}] + (\mathbb{E}_{\Theta_l}[\widehat{Y}^{MCMC}{}_{l,N_l}] - \mathbb{E}_{\nu^{l,l-1}}[\widehat{Y}^{MCMC}{}_{l,N_l}]) \leq CN_l^{-1}\mathbb{V}_{\nu^{l,l-1}}[Y_l]$, *and*
- $Cost(Q_l(\theta_l^i)) \leq CM_l^{\gamma}$.

Then there exist $L \in \mathbb{N}$ and $\{N_l\}_{l=0}^{L} \subset \mathbb{N}$, such that for the multilevel estimator's mean square error it holds $\mathbb{E}_{\cup_l \Theta_l}[(\widehat{Q}_{L,N_l}^{MLMCMC} - \mathbb{E}[Q])^2] < \epsilon^2$ and the overall cost of the multilevel estimator is given by

$$Cost(\widehat{Q}_{N_l,L}^{MLMCMC}) = C \left\{ \begin{array}{ccc} \mathcal{O}(\epsilon^{-2}|log(\epsilon)|) & for & \beta > \gamma, \\ \mathcal{O}(\epsilon^{-2}|log(\epsilon)|^3) & for & \beta = \gamma, \\ \mathcal{O}(\epsilon^{-2-\frac{\gamma-\beta}{\alpha}}|log(\epsilon)|) & for & \beta < \gamma. \end{array} \right\}.$$

Proof. See [Dod+19b, Theorem 3.4]. □

This cost estimate is now essentially analogous to the MLMC case in theorem 5, yielding similar gains in cost over the corresponding MCMC estimate

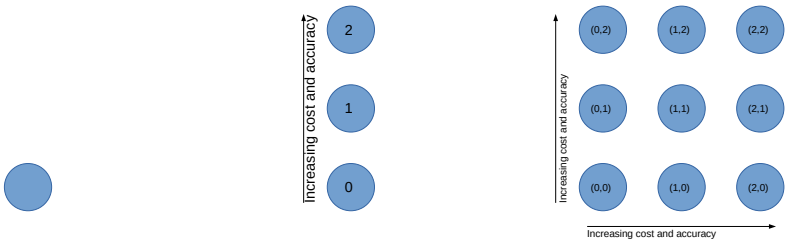
$$\mathcal{C}_\epsilon(\widehat{Q}_N^{MCMC}) \lesssim NM^\gamma \approx \epsilon^{-2-\gamma/\alpha}.$$

from section 2.4.3. Since posterior distributions vary across levels in the assumptions of theorem 7, applying it to our model problem from section 2.4.1 is not as straightforward as in the MLMC case above. For an application of theorem 7 to a realistic model, we therefore refer to [Dod+19b, Section 4.2], where it is rigorously shown that convergence an order below the single-level case can indeed be achieved.

3.4 Multiindex Markov Chain Monte Carlo

This section proposes a new extension of the above MLMCMC method to a more general multiindex setting.

Both MLMC and MLMCMC methods above exploit the fact that forward models can often be coarsened, in the sense of reducing computational effort while still achieving acceptable model predictions. There are, however, many forward models



(a) A single model as used in plain MCMC. **(b)** A linear model hierarchy as defined in MLMCMC. **(c)** A 2D model hierarchy for MIMCMC.

Fig. 3.3: Structures of model hierarchies from single level MCMC to MIMCMC.

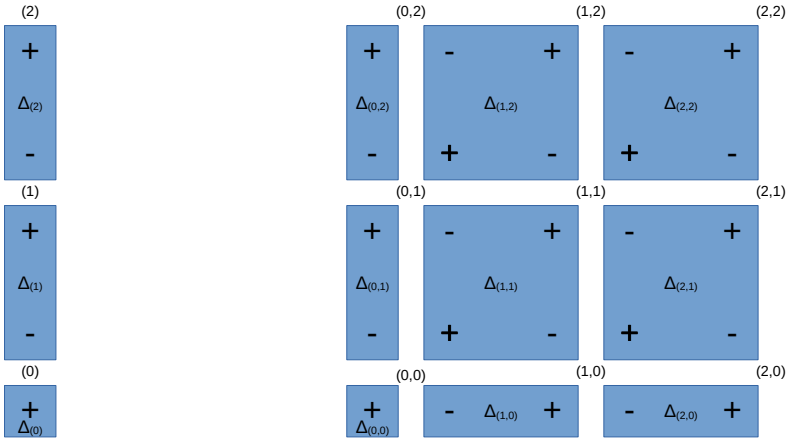
that can be coarsened in multiple independent ways. In order to exploit this, multiindex methods allow for an arbitrary dimensional "grid" of models instead of a "linear" multilevel model hierarchy. Each model is then identified by a multiindex $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$, where d is now the dimension of the model hierarchy. The resulting model hierarchies for single level, multilevel and multiindex methods are illustrated in fig. 3.3.

A generalization of MLMC to a multiindex setting, namely Multiindex Monte Carlo (MIMC), is proposed in [Haj+16]. Based on that, the authors of [Jas+17] show that a MCMC type method can be constructed in a multiindex setting as well. They do not, however, exploit coarser models as proposals. The goal of this section is therefore to transfer this key element from MLMCMC as constructed in [Dod+19b] to MIMCMC. Beyond efficiency gains due to cheap well-informed proposals, MIMCMC constructed in that manner is a true generalization of MLMCMC as introduced above. Therefore, a single implementation suffices in chapter 5 to cover both cases.

As a first step, we give a quick review of the key concepts in [Jas+17]. The notation from above is reused, the main difference is that objects defined per level are now defined per multiindex.

We again assume that the hierarchy of approximate models converges towards the underlying "true" model; the main difference is that now all dimensions of the multiindex need refinement:

$$\lim_{\min_{1\leq i\leq d} \alpha_i \to \infty} \mathbb{E}_{\nu^\alpha}[Q_\alpha] = \mathbb{E}_\nu[Q]$$



(a) Telescoping sum in multilevel case. **(b)** Telescoping sum in multiindex case.

Fig. 3.4: Structures of telescoping sums in MLMCMC and 2D MIMCMC case; the former is interpreted as a special case of the latter for one dimension. Each box represents a summand in eq. (3.8), i.e. a Δ operator applied to a specific multiindex. Each operator is ultimately comprised of summands itself according to eq. (3.6). A "+" symbol indicates a positive term in the operator, "-" indicates a negative sign. Note that, when adding all operators as in eq. (3.8), all but the finest models cancel out.

In order to construct a telescoping sum in the multiindex setting, we introduce the operators Δ_i, $1 \leq i \leq d$, defined in the following way:

$$\Delta_i \mathbb{E}_{\nu^\alpha}[Q_\alpha] \coloneqq \left\{ \begin{array}{cc} \mathbb{E}_{\nu^\alpha}[Q_\alpha] - \mathbb{E}_{\nu^{\alpha-e_i}}[Q_{\alpha-e_i}] & \text{if } \alpha_i > 0 \\ \mathbb{E}_{\nu^\alpha}[Q_\alpha] & else \end{array} \right\}, \tag{3.6}$$

where e_i are the canonical basis vectors of $\mathbb{R}^d$. Next, define the successive application of these operators as $\Delta \coloneqq \Delta_d \dots \Delta_1$. This leads us to the telescoping sum

$$\mathbb{E}_\nu[Q] = \sum_{\alpha \in \mathbb{N}_0^d} \Delta \mathbb{E}_{\nu^\alpha}[Q_\alpha] \tag{3.7}$$

As before, model evaluations for an infinite set of models are unavailable, and a truncation of the telescoping sum is used instead. Define a subset of $\mathbb{N}_0^d$

$$\mathcal{I}_{L_1:L_d} \coloneqq \{\alpha \in \mathbb{N}_0^d : \alpha_i \in \{0, \dots, L_i\} \forall i \in \{1, \dots, d\}\}.$$

The L_i correspond to the maximum number of "levels" available for the ith type of refinement. In case $d = 1$ and $L_1 = L$, this reduces to the multilevel setting treated above. Now the desired expected value is approximated by

$$\mathbb{E}_\nu[Q] \approx \sum_{\alpha \in \mathcal{I}_{L_1:L_d}} \Delta\mathbb{E}_{\nu^\alpha}[Q_\alpha]. \tag{3.8}$$

The construction of the telescoping sum eq. (3.8) is illustrated in fig. 3.4. Note that while here we choose the set $\mathcal{I}_{L_1:L_d}$ to form a hypercube grid, it was observed in the MIMC setting [Haj+16] that other "shapes" of index sets may lead to more efficient methods. In particular, a "convex" set is often a good choice since it may drop terms in the telescoping sum that are both very costly and do not contribute much to the error estimator due to their low variance.

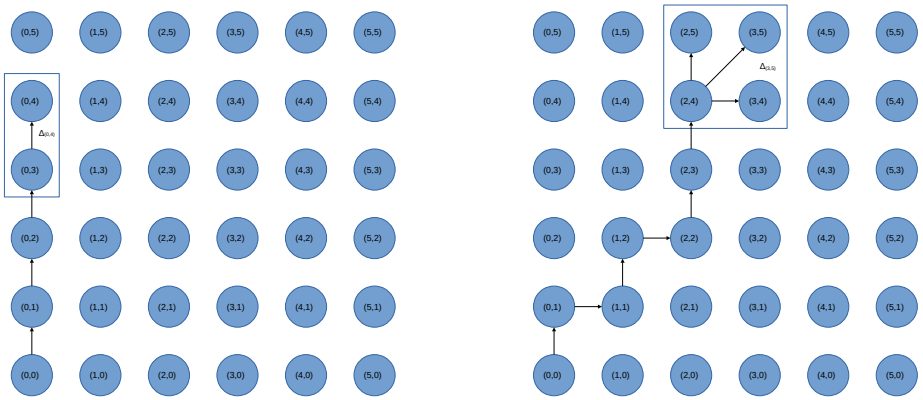
Algorithm 4: Multiindex MCMC

Result: Markov chains $\{\theta_k^i\}_{i=0}^{N_k}$ for all indices $k \in \mathcal{I}_{L_1:L_d}$.
On index $0 = (0, \ldots, 0)$, run a conventional MCMC chain, delivering samples θ_0^i.
for *index* $k \in \mathcal{I}_{L_1:L_d} \setminus 0$ **do**
  Choose starting point θ_k^0 with coarse component from next coarser starting point $\theta_{c(k)}^0$.
  for *sample* $j = 1, \ldots, N_k$ **do**
    Given θ_k^j, generate proposal $\theta_k' = \begin{Bmatrix} \theta_{k,C}' \\ \theta_{k,F} \end{Bmatrix}$ where
    - $\theta_{k,C}'$ is drawn from chain on level $c(k)$, observing subsampling, and
    - $\theta_{k,F}'$ from proposal density $q_k(\theta_{k,F}'|\theta_{k,F}^j)$.

    Compute acceptance probability

$$\alpha(\theta_k', \theta_k^j) = \min\left\{1, \frac{\pi_k(\theta_k')q_k(\theta_k^j|\theta_k')}{\pi_k(\theta_k^j)q_k(\theta_k'|\theta_k^j)} \cdot \frac{\pi_{c(k)}(\theta_{k,C}^j)}{\pi_{c(k)}(\theta_{k,C}')}\right\}.$$

    Draw a random number $r \in [0, 1]$.
    if $r < \alpha(\theta_k', \theta_k^j)$ **then**
      Accept proposal: θ_k' as θ_k^{j+1}
    else
      Reject proposal: $\theta_k^{j+1} = \theta_k^j$
    end
  end
end



(a) Proposal routing for $\Delta_{(0,4)}$.

(b) Proposal routing for $\Delta_{(3,5)}$.

Fig. 3.5: Routing of proposals in MIMCMC.

We now propose a new strategy for handling proposals in MIMCMC in analogy to the MLMCMC method from above. By coupling "neighboring" indices instead of levels, we can apply the same construction for sampling differences as already shown in algorithm 2. Specifically, we draw samples from coarser indices as proposals and, as before, choose a suitable acceptance ratio to ensure the fine chain still exhibits the appropriate stationary distribution. The result is an extension of algorithm algorithm 3 to its multiindex counterpart algorithm 4.

There is one significant difference compared to the MLMCMC variant: While the coarse level corresponding to level l is clearly level $l-1$, in the multiindex case it is not obvious which coarse chain index $c(\alpha) \in \mathbb{N}_0^d$ should provide proposals for chain $\alpha \in \mathbb{N}_0^d \setminus 0$.

The default choice taken in the implementation in chapter 5 is

$$c(\alpha) := (\alpha_1, \ldots, \alpha_{(argmax_i\ \alpha_i)-1}, \alpha_{(argmax_i\ \alpha_i)} - 1, \alpha_{(argmax_i\ \alpha_i)+1}, \ldots, \alpha_d), \quad (3.9)$$

where $\alpha \neq 0$. This is the most "direct" route towards the diagonal and then down to the coarsest index 0, while only coarsening one dimension at a time. This choice is not unique. A possibly more advanced option could be a dynamic choice minimizing cost along the path by balancing acceptance rates against model cost. This is however left to future investigations.

One further intricacy is that, in order to exploit variance reduction, samples used for estimating a difference in eq. (3.8) should be correlated across multiindices. In analogy to MLMCMC (see eq. (3.2)), we achieve this by using coarse samples as

proposals for fine chains and subsequently using fine samples and their corresponding coarse proposals together in the estimator. We therefore need to extend the aforementioned proposal routing strategy from eq. (3.9) to ensure that all samples used to estimate a difference in the telescoping sum (corresponding to a box in fig. 3.4) share proposals from the same coarse chain. The resulting routing strategy is illustrated in fig. 3.5.

Finally, in the case of a one-dimensional multiindex, the MIMCMC algorithm above is fully identical to MLMCMC: In case $d = 1$ the multiindex comes down to an integer and, with a proposal routing strategy defined as $c(\alpha) \coloneqq (\alpha_1 - 1)$, the telescoping sum eq. (3.8) collapses to eq. (3.1). Further, proposals are now simply drawn from the next coarser level again. This property allows a single implementation in chapter 5 to cover both multilevel and multiindex settings.

4 Efficient Models and Model Hierarchies for Multiscale Problems

Typical Uncertainty Quantification (UQ) problems as introduced in section 2.3 can be viewed as a deterministic forward model F combined with stochastic parameters, whose distribution is either prescribed in forward problems or to be determined from observations in inverse problems. This separation into a deterministic and a stochastic aspect immediately translates into two different approaches for improving the efficiency of numerical solvers for UQ problems: On one hand, we can improve the efficiency of the UQ method in the sense of reducing forward model evaluations, which has already been discussed extensively in chapter 3.

On the other hand, we can try and speed up the forward model itself. The efficient and scalable solution of deterministic forward models is of course a very active field of research on its own. For Partial Differential Equation (PDE) models alone, there is a wide variety of methods, typically in the form of efficient preconditioners, and software packages. Domain decomposition methods (see [SBG96] for an overview), multilevel methods like [SBG96] and various algebraic multigrid methods like BoomerAMG [YH02] or spectral AMGe [Cha+03] allow the solution of challenging large-scale models. They are available in software packages like DUNE [BHM10], FEniCS [Aln+15], Hypre [17] or deal.II [Alz+18]. These are just examples and far from an exhaustive list.

As such, we could focus entirely on the UQ aspect and simply rely on existing forward model implementations to be provided. However, there is a link between the two: Multilevel Markov Chain Monte Carlo (MLMCMC) and Multiindex Markov Chain Monte Carlo (MIMCMC) methods from chapter 3 act on a hierarchy of forward models. Since the real-world performance of those multilevel and multiindex methods depends entirely on the specific choice of that hierarchy just like multilevel forward solvers, it seems promising to investigate potential gains in employing purpose-built solvers for maximum efficiency of hierarchical UQ methods.

This chapter aims to investigate efficient choices for multilevel and multiindex model hierarchies for UQ methods by exploiting and extending results from modern

preconditioners for PDEs. We begin with a general discussion of model hierarchies in section 4.1.

In section 4.2 we present a new theoretical link between Generalized Eigenproblems in the Overlaps (GenEO), a robust and scalable preconditioner for PDEs, and an abstract Localized Model Order Reduction (LMOR) framework. This allows the exchange of methods between the two fields, which we demonstrate by applying a randomized eigensolver from LMOR in the preconditioner case. It also justifies the use of GenEO as a model order reduction method. This section is joint work with Andreas Buhr.

As an intermediate step in section 4.3, we proceed to present a new extension to our GenEO implementation in the Distributed and Unified Numerics Environment (DUNE) framework allowing the use of unstructured meshes. Support for unstructured meshes is particularly relevant for real-world engineering applications. This section is joint work with Peter Bastian. Peter Bastian provided the virtual overlap implementation, while the author was responsible for the GenEO implementation building on top of it.

We show in section 4.4 how to exploit the locality of the GenEO basis construction to form an efficient offline/online LMOR method that may reuse large parts of the expensive coarse basis across multiple evaluations. This section is joint work with Jean Bénézech. The author was mainly responsible for designing the method, while Jean Bénézech was mainly responsible for the implementation. This is an ongoing project, and the method will be applied to UQ problems in engineering in the near future.

4.1 Model Hierarchies

There is a number of different approaches to building model hierarchies suitable for multilevel and multiindex UQ methods. A frequently used example for forward models based on numerical PDE solvers is simply changing mesh width across levels. The main advantage is that numerical PDE theory typically provides a priori error estimates depending on mesh width, which allows the construction of a mesh hierarchy satisfying the assumptions on approximation quality made by e.g. MLMCMC theory as shown in section 3.3. The same holds for the multiindex case as well, as shown for a multiindex method in [Haj+16].

However, multilevel and multiindex methods as in chapter 3 only assume a hierarchy of posterior densities. Therefore we may also introduce a more general model hierarchy beyond mesh widths. For example, we might use a simple Ordinary Differential Equation (ODE) as a coarse approximation to a PDE model. A more in-depth treatment of such hierarchies is given under the name of multifidelity methods in [PWG16]. In our tsunami application in section 6.3, we use a smoothed bathymetry on coarser levels and even an extremely rough model without any bathymetry data. We complement this with mesh coarsening. Since we technically solve strongly modified models, we only obtain only very roughly correct outputs. However, as the solver for hyperbolic PDEs we use is strongly affected by wetting and drying as well as steep gradients in bathymetry, we gain significant reductions in computational cost for these simplified models. Even though a theoretical treatment of the approximation error we introduce in the process would be very hard to obtain, we clearly observe significant gains at the very least by achieving well-informed proposals for our finer models at low cost.

Beyond hand-crafted model hierarchies using application-specific knowledge, generating coarser models in an automatic way without requiring user intervention is an attractive proposition. In the context of PDE models, the field of Model Order Reduction (MOR) provides methods for automatically generating low-dimensional approximations to high-dimensional PDE discretizations. Markov Chain Monte Carlo (MCMC) type UQ methods require a large number of subsequent model evaluations for (in case of high autocorrelation) only sightly changed parameters. LMOR [Buh19] is of particular interest here, because local parameter changes between evaluations imply that only few parts of the reduced order model need to be updated each time.

In the following we prove that GenEO, a highly robust preconditioner, can be employed as an LMOR method as well. This allows us to use it in all stages of model hierarchy: On finer levels it serves as a preconditioner to an iterative solver for the discretized PDE. On the coarsest level in turn, we employ the GenEO coarse space as an LMOR method, reusing basis components from previous runs in an offline/online fashion and thereby essentially reducing cost to a few subdomain solves per model evaluation.

4.2 Linking Robust Preconditioners and Model Order Reduction

4.2.1 Efficient Preconditioning using GenEO coarse spaces

We show the link between model order reduction and eigenproblem-based robust preconditioning at the example of the GenEO (Generalized Eigenproblems in the Overlaps) coarse space. This section gives a brief review of the method as derived from abstract Schwarz preconditioning theory, largely based on the presentations in [SRS20; Spi+14].

Efficient preconditioning is crucial when solving large linear systems arising from finite element-type discretizations, as direct solvers easily become intractable and iterative solvers' performance directly depends on the condition number.

In order to exploit modern high performance computing architectures, a preconditioner should expose good parallel scalability. Further, solving hard simulation problems often requires robustness with respect to parameter contrast, anisotropies etc.

An established domain-decomposition approach achieving this is the abstract two-level Schwarz framework [SBG96; TW05]. We begin with a review of the underlying components and then proceed to show GenEO as a particular robust coarse space choice.

Abstract Two-Level Additive Schwarz

According to established Finite element (FE) theory, we assume a weak form of the PDE to be solved. As already introduced in section 2.1, that means we look for a solution $u \in U$ such that

$$a(u, v) = b(v) \qquad \forall v \in V. \tag{4.1}$$

For simplicity, we assume $U = V$, as is sufficient for many applications. The specific bilinear form a and linear form b arise in transforming a given PDE into such a weak form. Typically, the function space V will be a suitable Sobolev space defined on the PDE domain Ω, ensuring uniqueness of u.

Let $\mathcal{T}_h$ be a mesh of width h in the established FE sense. As a preparation for domain decomposition, we further make the assumption that a is not only symmetric positive definite but may also be decomposed into per-element weak forms

$$a(u,v) = \sum_{e \in \mathcal{T}_(} a_e(u,v).$$

This is trivially the case when a and b are integrals over Ω. For unions $\tilde{\Omega}$ of elements, we accordingly denote by $a_{\tilde{\Omega}}$ the sum of the corresponding a_e. We denote norms and semi-norms corresponding to $a_{\tilde{\Omega}}$ by $|\cdot|_{a,\tilde{\Omega}}$ and by $\|\cdot\|_{a,\tilde{\Omega}}$ in the respective cases.

In order to discretize, we again introduce a finite-dimensional approximation space $V_h \subset V$ based on per-element basis functions on the mesh $\mathcal{T}_h$, i.e. an FE space. The system matrix $\boldsymbol{A}$ is then a discrete representation of a in that space, defined by $\boldsymbol{A}_{i,j} = a(\phi_j, \phi_i)$ where $V_h = span\{\phi_1, \ldots, \phi_N\}$. Matrices and vectors used in the following can be determined analogously on the same basis.

This very brief introduction should suffice to establish notation. For a more detailed introduction to FE, refer to section 2.1.

The construction of a Schwarz type preconditioner now begins with a decomposition of the domain Ω into non-overlapping subdomains $\{\Omega_j^{'}\}_{j=1}^N$, where the $\Omega_j^{'}$ are unions of mesh elements of $\mathcal{T}_h$.

Next, we extend each non-overlapping domain by a layer of mesh elements, leading to now overlapping subdomains Ω_j. The extension procedure can be repeated an arbitrary number of times to achieve larger overlaps. We denote the overlap region of a subdomain as

$$\Omega_j^o := \{x \in \Omega_j : \exists j' \neq j \text{ such that } x \in \Omega_{j'}\},$$

which we will use during the construction of the coarse space.

We can define the according restrictions of our function space V_h onto the overlapping subdomains by $V_h(\Omega_j) := \{v|\Omega_j : v \in V_h\}$ for each $1 \leqslant j \leqslant N$. Further restricting these subspaces to homogeneous Dirichlet conditions on subdomain boundaries (not including $\partial\Omega$) we arrive at subspaces $V_{h,0}(\Omega_j)$.

Definition 26. Now define the prolongation operators $R_j^T : V_{h,0}(\Omega_j) \to V_h$ for each $v_j \in V_{h,0}(\Omega_j)$ such that $R_j^T v_j|\Omega_j = v_j$ and $R_j^T v_j|\Omega\backslash\Omega_j = 0$. The corresponding

restriction operator by R_j is then simply its adjoint. When representing the operator in matrix form, we use the notation $\boldsymbol{R}_j$.

Definition 27 (One-level Additive Schwarz)**.** Based on these restrictions and the system matrix $\boldsymbol{A}$, we can now define restricted system matrices $\boldsymbol{A}_j \coloneqq \boldsymbol{R}_j \boldsymbol{A} \boldsymbol{R}_j^T$. for all subdomains $j = 1, \dots, N$. Then the Additive Schwarz preconditioner is given by adding all local solves:

$$\boldsymbol{M}_{AS,1}^{-1} \coloneqq \sum_{j=1}^{N} \boldsymbol{R}_j^T \boldsymbol{A}_j^{-1} \boldsymbol{R}_j.$$

While such a preconditioner already is highly parallelizable, smooth errors spanning the entire domain are badly resolved by only local corrections. A typical approach to compensate for this is to introduce a coarser level to the method.

Following the one-level construction, we now introduce a coarse space V_H defined on the entire domain but with low dimension.

Definition 28 (Two-level Additive Schwarz)**.** As our coarse prolongation, denote the natural embedding by $R_H^T : V_H \to V_h$, and its matrix form $\boldsymbol{R_H}^T$. Again, the corresponding restriction is its adjoint R_H.

Now introducing the coarse problem $\mathbf{A}_H \coloneqq \boldsymbol{R}_H \boldsymbol{A} \boldsymbol{R}_H^T$, leads to the two-level Additive Schwarz preconditioner:

$$\boldsymbol{M}_{AS,2}^{-1} \coloneqq \boldsymbol{R}_H^T \boldsymbol{A}_H^{-1} \boldsymbol{R}_H + \sum_{j=1}^{N} \boldsymbol{R}_j^T \boldsymbol{A}_j^{-1} \boldsymbol{R}_j.$$

The effectiveness of such a preconditioner can be analyzed in the framework of [TW05]. This allows for an upper bound on the condition number of the preconditioned system, based on two essential properties which allow bounds on the smallest and largest eigenvalue.

The first property requires that only a certain number of subdomains may overlap at any given point.

Definition 29 (Covering bound)**.** We define

$$k_0 = \max_{\tau \in \mathcal{T}_h} (\#\{\Omega_j : 1 \leqslant j \leqslant N, \tau \subset \Omega_j\}).$$

Note that this property is usually easy to fulfill with a low constant, since it only depends on a reasonable arrangement of subdomains.

Lemma 5 (Upper bound)**.** *With a finite covering k_0, the largest eigenvalue of the preconditioned system can be bounded by*

$$\lambda_{\max}(\boldsymbol{M}_{AS,2}^{-1}\boldsymbol{A}) \leqslant k_0 + 1.$$

Proof. See [Spi+14, Lemma 2.6]. □

The second property requires the projections of a function onto the various subspaces are bounded by its norm.

Definition 30 (Stable splitting)**.**

$$\|R_H v\|_a^2 + \sum_{j=1}^{N} \|R_j v\|_{a,\Omega_j}^2 \leqslant C_0^2 \|v\|_a^2.$$

From this, we can conclude the desired bound on the smallest eigenvalue.

Lemma 6 (Lower bound)**.** *If there exists a stable splitting with a common constant C_0 for all $v \in V_h$, then it holds*

$$\lambda_{\min}(M_{AS,2}^{-1}A) \geqslant C_0^{-2}.$$

Proof. See [Spi+14, Theorem 2.8]. □

Putting all this together, we arrive at a condition bound.

Theorem 8. *Given Lemmas 5 and 6, we can give a bound on the condition number by*

$$\kappa(M_{AS,2}^{-1}A) \leqslant C_0^2(k_0 + 1).$$

Proof. See [Spi+14, Theorem 2.8]. □

While a good k_0 is easy to achieve, C_0 remains to be determined. In particular, while the restrictions R_j are obvious, the particular choice of R_H allows for some freedom. A large space V_H tends to lead to a more effective preconditioner, however implying a costly coarse system solve. Smaller coarse spaces are faster to solve in, but less of an acceleration to the method. Therefore, a good balance between the two must be found.

Additive Schwarz with GenEO Coarse Space

One such choice of coarse space is GenEO. If defines an efficient space with a good C_0 stable splitting constant by means of an eigenproblem. Here we show the definition of the method as well as the main theoretical results proving the robustness when applied in a preconditioner.

According to [Spi+11; Spi+14], Lemma 6 can be relaxed to inequalities of purely subdomain-local norms, albeit at the cost of a slightly worse constant:

Lemma 7 (Localized splitting)**.** *Let there exist a constant C_1 fulfilling*

$$\|R_j v\|^2_{a,\Omega_j} \leqslant C_1 |v|^2_{a,\Omega_j}$$

for all $1 \leqslant j \leqslant N$. Then the decomposition $v = R_H v + \sum_{j=1}^{N} R_j v$ as above is C_0-stable with $C_0^2 = 2 + C_1 k_0 (2k_0 + 1)$.

Note that these inequalities only involve the local restrictions onto V_j and not the coarse component V_H. Therefore, C_1 can now be optimized by writing both sides as part of an eigenproblem. Moving the eigenvectors implying a large C_1 into the coarse space consequently eliminates the respective eigenvalues from the inequalities.

Definition 31. For each subdomain $j = 1, \ldots, N$, we define the generalized eigenproblem: Find $p \in V_h(\Omega_j)$ such that

$$a_{\Omega_j}(p, v) = \lambda a_{\Omega_j^o}(\Xi_j(p), \Xi_j(v)), \quad \forall v \in V_h(\Omega_j). \tag{4.2}$$

Here, the Ξ_j denote a partition of unity according to the following definition.

Definition 32 (Partition of unity)**.** Let $dof(\Omega_j)$ denote the degrees of freedom on subdomain Ω_j. Given weights $\mu_{j,k} \geqslant 1$ with $\sum_{k \in dof(\Omega_j)} \frac{1}{\mu_{j,k}} = 1$ for each subdomain $1 \leqslant j \leqslant N$, the associated partition of unity operator is defined by

$$\Xi_j(v) \coloneqq \sum_{k \in dof(\Omega_j)} \frac{1}{\mu_{j,k}} v_k \phi_k | \Omega_j, \text{ for any } v \in V_h(\Omega_j).$$

This partition can be used to 'stitch together' local basis functions in order to produce a well-formed global basis.

By means of a partition of unity, the results of the eigenproblem in eq. (4.2) are now used to define the GenEO coarse space.

Definition 33 (GenEO coarse space)**.** For all subdomains $j = 1, \dots, N$, let $(p_k^j)_{k=1}^{m_j}$ denote the eigenfunctions from eigenproblem (4.2) corresponding to the m_j smallest eigenvalues. Then the GenEO coarse space is defined as

$$V_H \coloneqq span\{R_j^T \Xi_j(p_k^j) : k = 1, \dots, m_j; j = 1, \dots, N\}.$$

What remains is the choice of the numbers m_j of eigenvectors per subdomain to be included in the coarse space. By choosing all eigenvalues below a threshold, in [Spi+14] the following bound is proven.

Theorem 9. *For subdomain $1 \leqslant j \leqslant N$, choose*

$$m_j \coloneqq \min \left\{ m : \lambda_{m+1}^j > \frac{\delta_j}{H_j} \right\}$$

eigenvectors to be included in the coarse space, where δ_j is the overlap diameter Ω_j^o and $H_j = diam(\Omega_j)$. Then the condition number is bounded by

$$\kappa(\boldsymbol{M}_{AS,2}^{-1}\boldsymbol{A}) \leqslant (1 + k_0)\left[2 + k_0(2k_0 + 1) \max_{1 \leqslant j \leqslant N} \left(1 + \frac{H_j}{\delta_j}\right)\right].$$

Proof. See [Spi+14, Corollary 3.23]. □

This bound is now independent of parameter contrast and number of subdomains. Therefore, choosing a GenEO space in a two-level Additive Schwarz method promises to provide robustness and parallel scalability. These properties can be achieved in practice, as show in [But+20a; SRS20].

In software implementations, we solve the discrete form of the GenEO eigenproblem (4.2).

Definition 34. The discrete form of the GenEO eigenproblem for subdomain j reads

$$\tilde{\boldsymbol{A}}_j p_k^j = \lambda_k^j \boldsymbol{X}_j \tilde{\boldsymbol{A}}_j^o \boldsymbol{X}_j p_k^j.$$

Here we denote the subdomain-local discretization matrix by $\tilde{\boldsymbol{A}}_j$, the same matrix restricted to the overlap region by $\tilde{\boldsymbol{A}}_j^o$ and $\boldsymbol{X}_j$ is the discretized partition of unity (simply a diagonal matrix). In contrast to $\boldsymbol{A}_j$ from above, both $\tilde{\boldsymbol{A}}_j$ and $\tilde{\boldsymbol{A}}_j^o$ do not incorporate Dirichlet conditions away from $\partial\Omega$.

As it turns out, the GenEO theory requires only minor adjustments to also hold for a modified eigenproblem with full a full bilinear form on the right hand side. This modification plays an important role when we later show how to embed GenEO in the abstract LMOR framework, since it does not potentially introduce an additional kernel on the right hand side.

Definition 35. For each subdomain $j = 1, \dots, N$, we define the GenEO eigenproblem with full right hand side: Find $p \in V_h(\Omega_j)$ such that

$$a_{\Omega_j}(p, v) = \lambda a_{\Omega_j}(\Xi_j(p), \Xi_j(v)), \quad \forall v \in V_h(\Omega_j). \tag{4.3}$$

For full overlap, the two eigenproblems are clearly identical. Otherwise, we can replace [Spi+14, Lemma 3.21] by a simplified version. When showing the localized splitting property in lemma 7, instead of splitting the local contributions $R_j v$ into an overlap region and interior part, we can directly apply [Spi+14, Lemma 3.20] to it and conclude

$$\|R_j v\|_{a,\Omega_j} \leq \frac{1}{\lambda^j_{m_j+1}} \|v\|^2_{a,\Omega_j}.$$

Finally, an almost identical condition number estimate follows:

$$\kappa(\boldsymbol{M}^{-1}_{AS,2}\boldsymbol{A}) \leqslant (1 + k_0)\left[2 + k_0(2k_0 + 1) \max_{1 \leqslant j \leqslant N} \left(\frac{H_j}{\delta_j}\right)\right].$$

As a result, we can employ a space closely related to GenEO in LMOR even in settings with less than full overlap.

4.2.2 Localized Model Order Reduction

The goal of numerical model order reduction is to find a lower-dimensional approximation of a high-dimensional discretized model. There is a wide variety of methods to generate such a reduced model. The family of methods we focus on here is Localized Model Order Reduction (see [Buh+19] for an overview), since that approach exposes striking similarities to domain decomposition preconditioners and can in fact be shown to replicate theoretical results of GenEO and others (see section 4.2.3).

For an abstract framework of model order reduction theory, we follow the presentation in [Buh19] throughout this section. In order to remain consistent, we use

the same notation as in the original literature here. Later we show how the objects introduced here can be identified with or connected to the ones introduced in the preconditioner setting in section 4.2.1.

Abstract Localized Model Order Reduction

The goal of localized model order reduction is to generate a reduced model space by first decomposing the global space V into several subspaces. Then, local approximation spaces are generated and finally combined to form a global approximation space. Just like in domain decomposition preconditioners, this breaks down the global problem into computationally tractable smaller problems which can easily be parallelized in order to exploit modern high performance computers.

We begin by defining an abstract localizing space decomposition.

Definition 36 (Localizing Space Decomposition)**.** A localizing space decomposition consists of

1. an open, bounded and connected domain $\Omega \subset \mathbb{R}^d$ with smooth boundary,
2. a Hilbert space V of functions on Ω which is a continuous or discrete function space,
3. N_{V_i} subdomains ω_i which form an overlapping or non overlapping domain decomposition of the global domain Ω,
4. N_{V_i} local spaces V_i which are subspaces of the global ansatz space V and satisfy
$$\sum_i V_i = V \tag{4.4}$$
and
$$\operatorname{supp}(\varphi) \subseteq \omega_i \qquad \forall \varphi \in V_i, \tag{4.5}$$
5. N_{V_i} linear mappings $\mathcal{P}_{V_i} : V \to V_i$ for which it holds
$$\sum_i \mathcal{P}_{V_i}(\varphi) = \varphi \qquad \forall \varphi \in V. \tag{4.6}$$

Instead of solving our variational problem from eq. (4.1) in the full space V, we will solve a reduced version in an approximating space.

Definition 37 (Reduced variational problem)**.** Find $\widetilde{u}$ in $\widetilde{V} \subset V$, such that

$$a(\widetilde{u}, v) = b(v) \qquad \forall v \in \widetilde{V}. \tag{4.7}$$

Based on the space decomposition above, we can now proceed to define local reduced spaces that will, when combined, form the global reduced space $\widetilde{V}$.

Definition 38 (Localized training configuration)**.** Let $\left\{\Omega, V, \{\omega_i\}_{i=1}^{N_{V_i}}, \{V_i\}_{i=1}^{N_{V_i}}, \{\mathcal{P}_{V_i}\}_{i=1}^{N_{V_i}}\right\}$ be a localizing space decomposition as defined in definition 36 and let u be the solution of a non parametric variational problem as defined in eq. (4.1). We define a "localized training configuration" to be a set of

1. N_{V_i} training domains $\omega_i^* \subset \Omega$ with the maximum number of subdomains overlapping at any point x of Ω denoted by

$$J^* := \max_{x \in \Omega} \#\{i \in \{1, \dots, N_{V_i}\} |\ x \in \omega_i^*\}, \tag{4.8}$$

2. N_{V_i} Hilbert spaces $V|_{\omega_i^*}$ equipped with a norm satisfying

$$\sum_{i=1}^{N_{V_i}} \left\| u|_{\omega_i^*} \right\|^2 \leq J^* \|u\|^2, \tag{4.9}$$

3. N_{V_i} Hilbert spaces S_i,
4. N_{V_i} bounded linear operators $\mathcal{P}_{S_i} : V|_{\omega_i^*} \to S_i$,
5. N_{V_i} affine linear and compact operators $T_i : S_i \to V_i$,

for which it holds

$$u = \sum_{i=1}^{N_{V_i}} T_i \mathcal{P}_{S_i} u|_{\omega_i^*}. \tag{4.10}$$

Further, denote the linear components and affine shifts of T_i by T_i^a and T_i^l respectively.

For such a localized training configuration, the following approximation result can be shown.

Theorem 10 (Training a priori estimate)**.** *Let* $\left\{\Omega, V, \{\omega_i\}_{i=1}^{N_{V_i}}, \{V_i\}_{i=1}^{N_{V_i}}, \{\mathcal{P}_{V_i}\}_{i=1}^{N_{V_i}}\right\}$ *be a localizing space decomposition as defined in 36. Let further* $\left\{\{\omega_i^*\}_{i=1}^{N_{V_i}}, \{V|_{\omega_i^*}\}_{i=1}^{N_{V_i}}, \{S_i\}_{i=1}^{N_{V_i}}, \{\mathcal{P}_{S_i}\}_{i=1}^{N_{V_i}}, \{T_i\}_{i=1}^{N_{V_i}}\right\}$ *be a localized training configuration as defined in definition 38. Let* u *be the solution of a symmetric, coercive and stable variational problem as defined in eq.* (4.1) *and let* $\widetilde{u}$ *be the reduced solution of a localized Galerkin projection onto* $\widetilde{V} = \sum_{i=1}^{N_{V_i}} \widetilde{V}_i$ *as introduced in definition 37. Further assume*

$$T_i^a \in \widetilde{V}_i \qquad \forall i \in \{1, \dots, N_{V_i}\}, \tag{4.11}$$

and assume that the spaces V_i *can be partitioned into* $\bar{J}$ *classes such that spaces within a class are orthogonal.*

Then it holds

$$\frac{\|u - \widetilde{u}\|}{\|u\|} \leq \sqrt{\frac{\gamma}{\alpha}} \sqrt{\bar{J} J^*} \max_{i \in \{1,\dots,N_{V_i}\}} \left\| \left(1 - P_{\widetilde{V}_i}\right) T_i^l \right\| \|\mathcal{P}_{S_i}\|. \tag{4.12}$$

Proof. See [Buh19, Proposition 5.1.3]. □

Optimal Spaces

Since the approximation result in theorem 10 is formulated in a general way, it remains to choose the localized training configuration optimizing approximation error for a given reduced space dimension.

According to [SP16a; BL11], for a given reduced space dimension n, the error $\left\| \left(1 - P_{\widetilde{R}^n}\right) T^l \right\|$ is minimized by the space spanned by the first n left singular vectors of T^l. The Singular Value Decomposition (SVD) exists due to the assumed compactness of T^l. We denote it by

$$T^l \varphi = \sum_{i=1}^{N_T} q_i \sigma_i (v_i, \varphi)_S \qquad \forall \varphi \in S, \tag{4.13}$$

where N_T is the rank of T^l and it holds $(q_i, q_j)_R = \delta_{ij}$, $(v_i, v_j)_S = \delta_{ij}$ and $\sigma_i \in \mathbb{R}^+$. The optimal spaces defined as

$$\widetilde{R}^n_{\text{opt}} := \text{span}\{q_1, \dots, q_n\} \tag{4.14}$$

then minimize $\left\| \left(1 - P_{\widetilde{R}^n}\right) T^l \right\|$ for the given space dimension n and were applied in [BL11] and [SP16b].

Lemma 8. *Exploiting properties of the SVD and space decomposition, it holds*

$$\begin{aligned}
\left\| \left(1 - P_{\widetilde{R}^n_{\text{opt}}}\right) T^l \right\| &= \sup_{\varphi \in S \setminus \{0\}} \frac{\left\| \left(1 - P_{\widetilde{R}^n_{\text{opt}}}\right) T^l \varphi \right\|}{\|\varphi\|} \\
&= \sup_{\varphi \in S \setminus \{0\}} \frac{\left\| \left(1 - P_{\widetilde{R}^n_{\text{opt}}}\right) \sum_{i=1}^{N_T} q_i \sigma_i (v_i, \varphi)_S \right\|}{\|\varphi\|} \\
&= \sup_{\varphi \in S \setminus \{0\}} \frac{\left\| \sum_{i=n+1}^{N_T} q_i \sigma_i (v_i, \varphi)_S \right\|}{\|\varphi\|} \\
&= \sigma_{n+1}. \qquad (4.15)
\end{aligned}$$

So, choosing the coarse subspace as the first n singular vectors of T_i^l, the error bound in theorem 10 reduces to the product of some problem-specific constants and the first singular value not included in the coarse space.

4.2.3 GenEO Coarse Space in LMOR Theory

In this section, we show how the GenEO coarse space can be covered in the abstract LMOR theory above. To that end, we use the modified GenEO space with a full right hand side of the eigenproblem defined in definition 35. Roughly speaking, the original GenEO coarse space is constructed with a two-level Schwarz setting in mind, and due to local subdomain solves the interior of subdomains does not need to be treated by the coarse space. For LMOR however, we need to represent the solution on the entire domain, and not just optimize the coupling between subdomains via overlaps.

Further, the abstract LMOR theory above requires some of the spaces involved to be Hilbert spaces. The bilinear forms originating from weak formulations of elliptic PDEs may however have a non-trivial kernel $ker(a)$, and they can therefore not serve as an inner product. To fix this, we only treat $V_h/ker(a)$ as the space to apply LMOR on, and later add the kernel to the coarse space again. Note that in practice this step can be ignored, since the GenEO eigenproblem recovers the kernel as eigenvectors with zero eigenvalue, and therefore the GenEO space as defined above always includes the kernel anyways.

Definition 39. In order to represent GenEO within the setting of theorem 10, we identify the following objects in LMOR by their respective counterparts in GenEO notation:

Domains		Spaces		Mappings		Inner products
LMOR	GenEO	LMOR	GenEO	LMOR	GenEO	
Ω	Ω	V	$V_h/ker(a)$			$a(\cdot,\cdot)$
ω_i	Ω_i	V_i	$V_{h,0}(\Omega_i)/ker(a)$	$\mathcal{P}_{V_i}$	Ξ_i	
ω_i^*	Ω_i	$V\|_{\omega_i^*}$	$V_h(\Omega_i)/ker(a)$			$a_{\Omega_i}(\cdot,\cdot)$
		S_i	$V_h(\Omega_i)/ker(a)$	$\mathcal{P}_{S_i}$	$\mathbb{1}$	$a_{\Omega_i}(\cdot,\cdot)$
				T_i	Ξ_i	

As the partition of unity Ξ_i is linear, we trivially have $T_i^l = T_i$ and $T_i^a = 0$

Lemma 9. *The domain decomposition used in GenEO is a localizing space decomposition as in definition 36.*

Proof. We use the identifications of definition 39 to show that definition 36 is fulfilled.

1. As defined in the GenEO setting, Ω is indeed an open, bounded and connected domain.
2. The Hilbert space V is chosen as the discrete space $V_h(\Omega)/ker(a)$ from the GenEO setting.
3. Local spaces V_i, identified with $V_{h,0}(\Omega_i)/ker(a)$ from GenEO, clearly fulfill $\sum_i V_i = V$ while consisting of functions with only local support.
4. When identifying the $\mathcal{P}_{V_i}$ with the partition of unity, this property is fulfilled by definition.

□

Lemma 10. *With choices of operators as in definition 39, GenEO is a Localized Training Configuration as in definition 38.*

Proof. 1. By definition $\kappa_0 = J^*$.

2. By finite covering, we trivially have

$$\sum_{i=1}^{N_{V_i}} |u|_{\omega_i^*}|_{a,\Omega_j}^2 \leq \kappa_0 \|u\|_{a,\Omega}^2. \tag{4.16}$$

3. ${}^{V_h(\Omega_j)}/_{ker(a)}$ equipped with the inner product a_{Ω_j} is a Hilbert space.

4. The operators $\mathcal{P}_{S_i} = \mathbb{1}$ are trivially bounded and linear.

5. Likewise, the operators $T_i = \Xi_i$ are linear. Due to the choice of $\mathcal{P}_{S_i}$ and by definition of the partition of unity, it holds

$$\sum_{i=1}^{N_{V_i}} T_i \mathcal{P}_{S_i} u|_{\omega_i^*} = \sum_{i=1}^{N_{V_i}} \Xi_i u|_{\Omega_j} = u. \tag{4.17}$$

   Since we are treating a finite dimensional setting, the operators are also compact.

□

We have now shown that, with the choice of spaces from definition 39, we fulfill the localized model order reduction assumptions from section 4.2.2. As a result, we obtain the optimal coarse approximation space $P_{R^n_{opt}}$ and coarse approximation error estimates due to lemma 8. It remains to show that this optimal space matches the corresponding GenEO space of the same dimension. To that end, it is enough to show that the per-subdomain GenEO eigenproblem from definition 35 is identical to the per-subdomain operator SVDs from localized model order reduction:

$$\begin{aligned}
& a_{\Omega_i}(p_j, v) = \lambda_j a_{\Omega_i}(\Xi_i(p_j), \Xi_i(v)) && \forall v \in V_h(\Omega_j) \\
\iff\ & a_{\Omega_i}(p_j, v) = \lambda_j a_{\Omega_i}(T_i(p_j), T_i(v)) && \forall v \in V_h(\Omega_j) \\
\iff\ & (p_j, v)_{S_i} = \lambda_j (T_i p_j, T_i v)_{V_i} && \forall v \in S_i.
\end{aligned}$$

Defining $\sigma_j = \frac{1}{\sqrt{\lambda_j}}$, we can formulate

$$T_i \phi = \sum_j \frac{1}{\sigma_j} T_i p_j \sigma_j (p_j, \phi)_{S_i}.$$

This is indeed the operator SVD of T_i, since from the eigenproblem we know $(p_j, p_k)_{S_i} = \delta_{ij}$ and $(T_i p_j, T_i p_k)_{V_i} = \lambda_j \delta_{jk}$, where the latter implies $(\frac{1}{\sigma_j} T_i p_j, \frac{1}{\sigma_k} T_i p_k)_{V_i} = \delta_{jk}$, as well as $\sigma_j > 0$.

The GenEO space is formed from vectors $\Xi_i p_j$ and the LMOR space R^n_{opt} from vectors $\frac{1}{\sigma_j} T_i p_j$, which is, up to a scaling factor, identical.

Finally, having restricted ourselves to ${}^{V_h}/_{ker(a)}$, we augment the LMOR space by the kernel. The resulting space $R^n_{opt} \bigoplus ker(a)$ serves as an approximation to the full discrete space V_h. Since we include the full kernel again, no additional approximation error is incurred.

As a further link between GenEO and LMOR, we additionally observe that, with the newly obtained approximation result, we may substitute part of the GenEO preconditioner analysis above.

Lemma 11. *Using the approximation property in lemma 8, we can prove the assumption of lemma 7.*

Proof.

$$
\begin{aligned}
\|z_j\|^2_{a,\Omega_j} &= \|\Xi_j(\mathbb{1} - \Pi^j_{m_j})v|_{\Omega_j}\|_{a,\Omega_j} \\
&= \|\Xi_j v|_{\Omega_j} - \Xi_j \sum_{k=1}^{m_j} a_{\Omega_j}(\Xi_j v|_{\Omega_j}, \Xi_j p^j_k) p^j_k\|_{a,\Omega_j} \\
&= \|\Xi_j v|_{\Omega_j} - \sum_{k=1}^{m_j} a_{\Omega_j}(\Xi_j v|_{\Omega_j}, \Xi_j p^j_k) \Xi_j p^j_k\|_{a,\Omega_j} \\
&= \|(\mathbb{1} - P_{GenEO})\Xi_j v|_{\Omega_j}\|_{a,\Omega_j} \\
&= \|(\mathbb{1} - P_{R^n_{opt}})T_j v|_{\Omega_j}\|_{a,\Omega_j} \\
&\leqslant \|(\mathbb{1} - P_{R^n_{opt}})T_j\|_{a,\Omega_j} \|v|_{\Omega_j}\|_{a,\Omega_j} \\
&\leqslant \sigma^j_{m_j+1} |v|_{a,\Omega_j}.
\end{aligned}
$$

□

With the implied stable splitting constant, we can conclude a condition number bound from theorem 8 yielding the same result as theorem 9:

$$
\kappa(\boldsymbol{M}^{-1}_{\mathrm{AS},2}\boldsymbol{A}) \leqslant (1 + k_0)\left[2 + k_0(2k_0 + 1) \max_{1 \leqslant j \leqslant N} \left(\sigma^j_{m_j+1}\right)\right].
$$

4.2.4 GenEO with Randomized Eigensolver

Randomized Eigensolver

Randomized techniques have seen large interest from the model order reduction community during recent years. The optimal spaces defined in Equation 4.14 can be approximated by randomized techniques at reduced computational cost, while limiting the approximation error with very high probability [BS18] [Buh19, Section 5.2].

Algorithm 5: Randomized GenEO eigensolver

Data: Matrices from GenEO eigenproblem $\tilde{\boldsymbol{A}}_j$, $\tilde{\boldsymbol{A}}_j^o$ and $\boldsymbol{X}_j$. Number of reiterations $R \in \mathbb{N}$
Result: Approximate GenEO eigenvectors $\{p_1, \ldots, p_k\}$.
Compute Cholesky factorization $\boldsymbol{C}\boldsymbol{C}^T = \tilde{\boldsymbol{A}}_j$.
for $i = 1, \ldots, k$ **do**
  Draw random vector $r \sim N(0, I)$.

$$p_i \coloneqq (\boldsymbol{X}_j \tilde{\boldsymbol{A}}_j^{-1} \boldsymbol{X}_j \boldsymbol{X}_j \tilde{\boldsymbol{A}}_j^o \boldsymbol{X}_j)^R \boldsymbol{X}_j \boldsymbol{C}^{-T} r$$

end
Orthogonalize $p_1, ..., p_k$ w.r.t. $\boldsymbol{X}_j \tilde{\boldsymbol{A}}_j^o \boldsymbol{X}_j$.

We adapt the procedure in [Buh19, Algorithm 5.1] to the GenEO case and simplify it to our needs in algorithm 5. The resulting algorithm allows tweaking the eigenvectors' accuracy via the number of reiterations. However, compared to the original, we drop the test vector approach to verifying approximation quality, since it turned out too costly in our applications. As we will later investigate at numerical examples, coarse space quality is far less relevant for overall solver performance in the preconditioning case than it clearly would be in LMOR applications.

Compared to purely iterative eigensolvers, algorithm 5 still has one significant disadvantage in that it requires a costly Cholesky factorization. This turns out prohibitive in practice, so for the subsequent numerical experiments we allow ourselves to instead use the root of the diagonal as a crude approximation to the Cholesky inverse. The result is algorithm 6. Unfortunately this is not backed by the theoretical framework in [Buh19]. It is however a reasonable approximation in the sense that the Cholesky inverse is only needed to transform the randomized starting vectors, so we should only lose some efficiency in coarse space approximation and not systematically miss important eigenvectors. This is supported by the numerical experiments showing competitive behavior compared to an iterative eigensolver.

Algorithm 6: Fast Randomized GenEO eigensolver

Data: Matrices from GenEO eigenproblem $\tilde{\boldsymbol{A}}_j$, $\tilde{\boldsymbol{A}}_j^o$ and $\boldsymbol{X}_j$. Number of reiterations $R \in \mathbb{N}$

Result: Approximate GenEO eigenvectors $\{p_1, \ldots, p_k\}$.

for $i = 1, \ldots, k$ **do**

Draw random vector $r \sim N(0, I)$.

$$p_i := (\boldsymbol{X}_j \tilde{\boldsymbol{A}}_j^{-1} \boldsymbol{X}_j \boldsymbol{X}_j \tilde{\boldsymbol{A}}_j^o \boldsymbol{X}_j)^R \boldsymbol{X}_j diag(\tilde{\boldsymbol{A}}_j)^{-1/2} r$$

end

Orthogonalize $p_1, ..., p_k$ w.r.t. $\boldsymbol{X}_j \tilde{\boldsymbol{A}}_j^o \boldsymbol{X}_j$.

Implementation

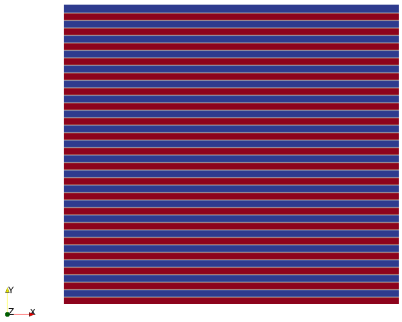
For the linear elasticity model, we use the the package `dune-composites` [But+20b], which is primarily intended for modeling composite materials. It has successfully been applied in real-world applications, e.g. in [Rei+18].

The GenEO implementation we use has been introduced in [SRS20]. It is part of `dune-pdelab` and also serves as the main solver for `dune-composites`. In the framework of this GenEO code, we implemented the randomized GenEO solver introduced in section 4.2.4 as an alternative local basis. This is achieved by subclassing `SubdomainBasis`. The result can easily be used in place of the existing GenEO local basis implementation based on ARPACK [LSY97] when desired. The remainder of the framework, which among others constructs a global basis etc., remains untouched.

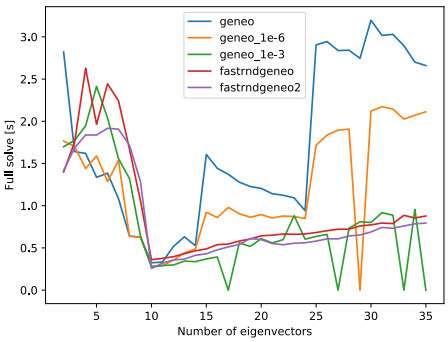
Poisson

For the Poisson equation tests, we choose a parameter distribution for $\kappa(x)$ as shown in Fig. 4.1a with a contrast of 10^5. As indicated by the results in Fig. 4.1d, computing the per-subdomain basis vectors turns out to be considerably faster with the randomized approach compared to the iterative one based on ARPACK. In particular, setup cost is independent of the eigenproblem's spectrum.

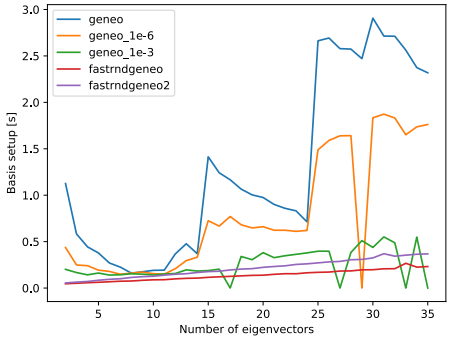
The gain is somewhat offset by an increased iteration number of the CG method (Fig. 4.1e), indicating a lower-quality basis as expected. For extremely low numbers of eigenvectors, that effect becomes very apparent, as the first eigenvectors tend to have a very prominent effect. Still, for reasonable basis sizes, the randomized method outperforms the iterative one in total time (Fig. 4.1c). Note that increasing



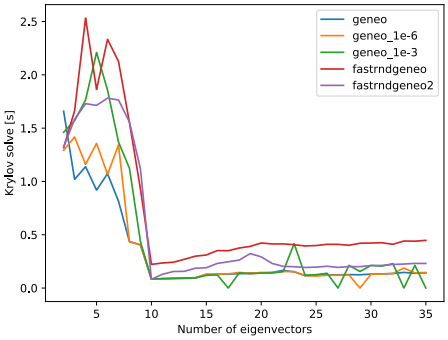
(a) Parameter distribution used, where blue is low and red is high permeability.



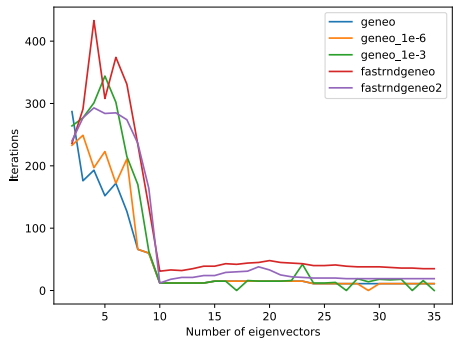
(b) Total time for preconditioner setup and preconditioned solver.



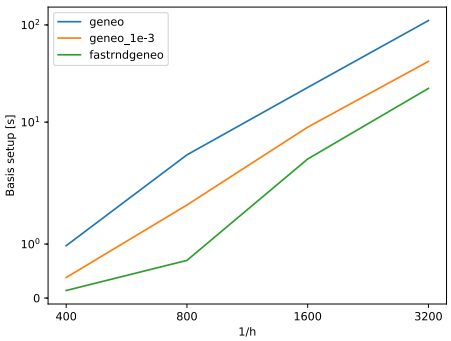
(c) Setup time for per-subdomain basis.



(d) Time taken by preconditioned CG solver.

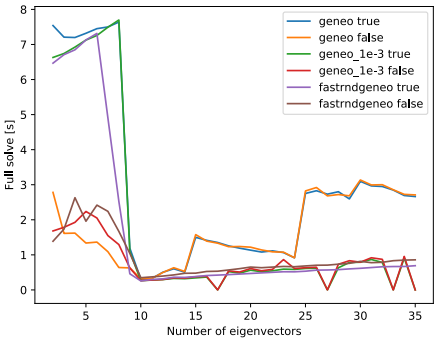


(e) Iteration numbers of preconditioned CG.

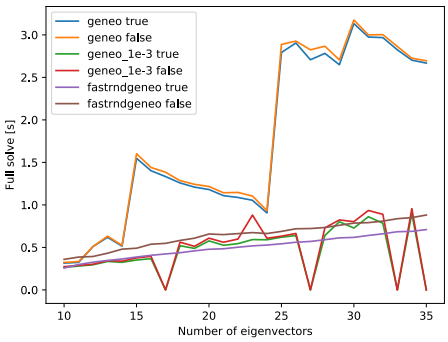


(f) Basis setup cost for decreasing element size h when computing 15 basis vectors.

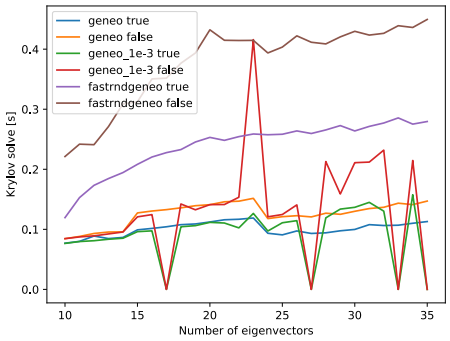
Fig. 4.1: Results for Poisson application. Failed solves due to ARPACK crashes are indicated by zero values. *geneo* refers to eigensolves using ARPACK to full accuracy, *geneo_1e-6* and *geneo_1e-3* introduce the respective error tolerance. *fastrndgeneo* is the randomized method with a single reiteration and *fastrndgeneo2* adds a second reiteration.



(a) Total time for solve.



(b) Total time for solve ($\geqslant$ 10 EV).



(c) Time spent in preconditioned Krylov solver ($\geqslant$ 10 EV).

Fig. 4.2: Results comparing restricted hybrid (*true*) to the additive (*false*) method in our Poisson test case. Results are shown for various basis setup methods as above.

the error tolerances for ARPACK leads to a performance rivaling the randomized approach. However, this leads to instabilities as ARPACK may return a near-singular basis in that case.

Further, applying two reiterations in the randomized method instead of one leads a more accurate eigensolve. This implies slightly higher setup cost while offering slightly lower iterations numbers. Overall, we achieve a minor gain here.

Concerning scalability, basis setup cost scales comparably for iterative and randomized eigensolves with respect to numbers of degrees of freedom (fig. 4.1f).

Finally, a hybrid additive/multiplicative two-level Schwarz method (fig. 4.2) speeds up the Krylov solve considerably for less exact methods (fig. 4.2c), in particular for the randomized one. Note that, for very small numbers of eigenvectors, the insufficient coarse space leads to an even larger loss of performance for hybrid methods than is the case for fully additive ones.

Since this introduces asymmetry, we use GMRES instead of CG as our solver here. Therefore, we can additionally use restricted additive Schwarz (RAS) [CS99] in these cases, employing the partition of unity defined above.

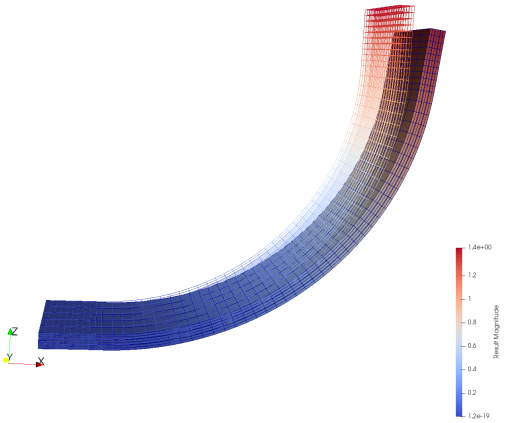
The restricted and hybrid two-level additive Schwarz method can be defined by extending Def. 28 in the following way:

$$\boldsymbol{M}_{hAS,2}^{-1} \coloneqq \boldsymbol{R}_H^T \boldsymbol{A}_H^{-1} \boldsymbol{R}_H (I - \boldsymbol{A} \sum_{j=1}^{N} \boldsymbol{R}_j^T \boldsymbol{X}_j \boldsymbol{A}_j^{-1} \boldsymbol{R}_j) + \sum_{j=1}^{N} \boldsymbol{R}_j^T \boldsymbol{X}_j \boldsymbol{A}_j^{-1} \boldsymbol{R}_j.$$

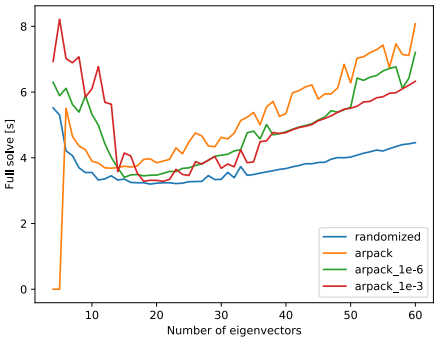
This modification introduces only very little overhead, as the per-subdomain solves still only have to be performed once in practice and applying a partition of unity is cheap.

Linear Elasticity

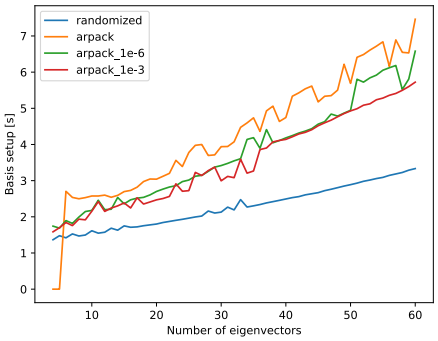
In case of the Linear Elasticity equation, we choose a model similar to the ones presented in [Rei+18]. The geometry is depicted in Fig. 4.3a. As boundary conditions, zero Dirichlet conditions are applied on the left side, and a force pulling to the right is applied at the top. The material is comprised of three highly anisotropic layers of carbon fiber sheets in 45, -45 and 0 degrees rotation as well as two isotropic resin layers in between.



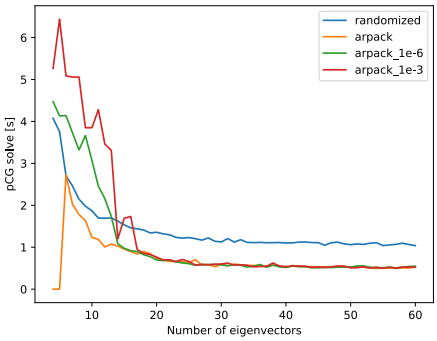
(a) Original geometry as mesh and displacement solution as solid volume (displacement scaled by factor 5 for better visibility).



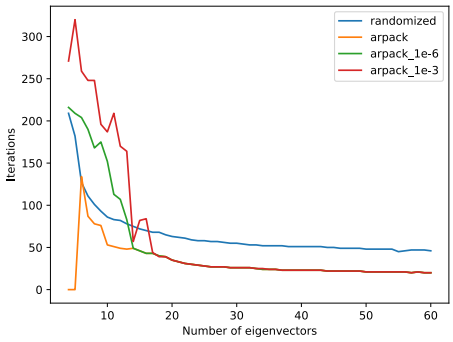
(b) Total time for preconditioner setup and preconditioned solver.



(c) Setup time for per-subdomain basis.

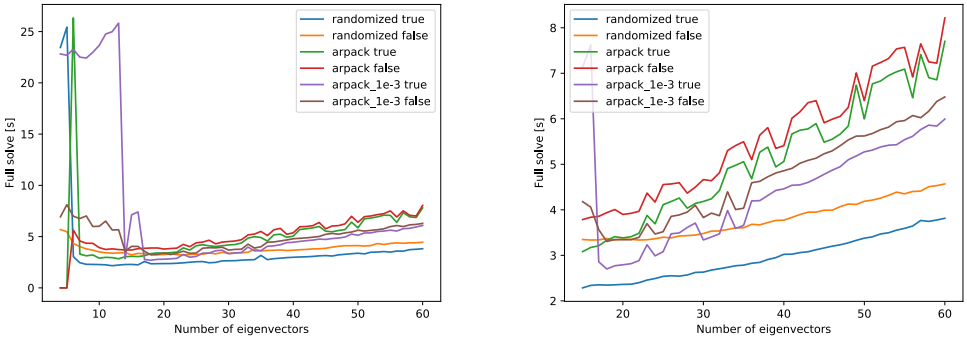


(d) Time taken by preconditioned CG solver.



(e) Iteration numbers of preconditioned CG.

Fig. 4.3: Results for Linear Elasticity application. Failed solves are indicated by zero values.



(a) Total time for preconditioner setup and preconditioned solver.

(b) Detailed view of Fig. 4.4a for EV >= 15.

Fig. 4.4: Restricted hybrid additive Schwarz for Linear Elasticity application.

When testing both approaches to solving the eigenproblem, a similar picture emerges, see Fig. 4.3. A significantly lower basis setup time in the randomized case is partly offset by slightly higher iteration numbers, but overall still leads to an improved total time. In particular, the randomized solver tends to depend less on the particular choice of basis size. Higher error tolerances lead to a faster ARPACK solve at the cost of a lower quality basis and therefore a slower preconditioner.

When applying the same restricted and hybrid additive Schwarz method here, we again see a significant improvement (Fig. 4.4). Now, the randomized method is leading clearly over the iterative ones except for unrealistically small numbers of eigenvectors.

4.3 GenEO Virtual Overlap Implementation

The DUNE framework provides a number of grid implementations for various purposes. Currently only `YASPGrid`, a structured grid, provides native support for subdomain overlaps. This means that each process holds a copy of elements in its overlap region and may exchange data attached to associated Degree of freedom (DOF)s with neighboring processes. Since GenEO is based on overlapping subdomains, we need this kind of communication mechanism. In previous work [Rei+18], `dune-composites` overcame the restriction to cuboid domains implied by `YASPGrid` by applying a transforming to the grid. However, for many relevant engineering applications, a smooth transition from a cuboid cannot be found and we need to extend DOF-based communication support unstructured grids as well.

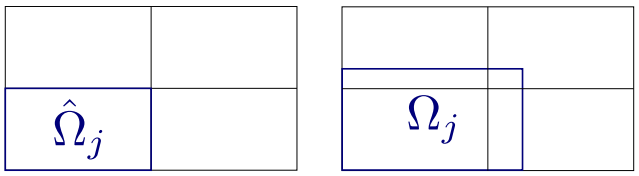


Fig. 4.5: Definition of subdomains used in the following.

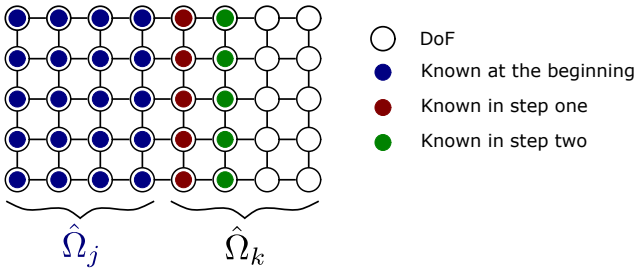


Fig. 4.6: The degrees of freedom known to process j after each step of recursive extension of the matrix connectivity graph.

In order provide overlaps and communication across overlaps on unstructured grids as well, we introduce an algebraic approach to construct discretization matrices A_j defined overlapping subdomains Ω_j from corresponding matrices $\hat{A}_j$ assembled on non-overlapping subdomains $\hat{\Omega}_j$. Figure 4.5 illustrates our notation for non-overlapping and overlapping subdomains.

We begin with a matrix $\hat{A}_j$ on a non-overlapping subdomain $\hat{\Omega}_j$, and assume it has a neighboring subdomains $\hat{\Omega}_k$, $k \in E_j$, with corresponding matrices $\hat{A}_k$. We identify coinciding DOFs in $\hat{A}_j$ and $\hat{A}_k$ through a global indexing as provided by non-overlapping DUNE grids. On subdomain k, we now identify all DOFs directly connected to those, and provide process j with unique indices and the connectivity graph of this newly identified layer of DOFs. Since connected DOFs in a FE discretization either share an element or are from adjacent elements, we now are in a state where each process is aware of its neighbors' DOFs one layer of elements along the respective boundary.

By recursive application as shown in fig. 4.6, we can now grow this algebraic overlap by an arbitrary number of layers, without having to rely on the grid implementation to provide connectivity graphs.

Finally, using the extended connectivity graphs above we can construct a communication mechanism to exchange overlap data of vectors defined on the extended domains between neighbors. We now have the communication infrastructure necessary to generate a GenEO space in a parallel method in place.

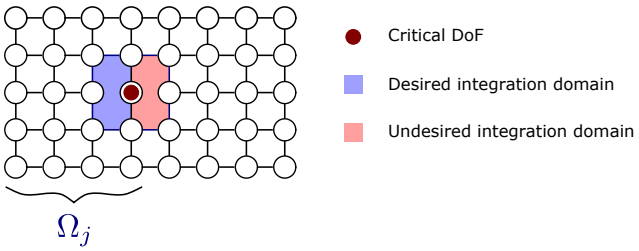


Fig. 4.7: Integration domain of a DOF on the boundary of the overlap of Ω_j. Since it lies in the interior of a neighboring non-overlapping domain, its matrix entries are computed by the neighbor. The neighbor takes the entire integration domain around the DOF into account, while process j requires only integration up to the overlapping subdomain boundary $\partial\Omega_j$. The neighbors' matrix entries can therefore not be used to assemble A_j.

While the method above allows us to extend connectivity graphs of non-overlapping matrices into neighboring subdomains, mimicking the graphs of overlapping matrices A_j, the GenEO method obviously requires the actual matrix entries. Using communication across algebraic overlap, we can exchange matrix entries between neighbors.

The basic assumption we make here is that the bilinear form a of the weak formulation can be decomposed additively into bilinear forms on elements a_ϵ, such that $a(u,v) = \sum_{\epsilon\in\mathcal{T}_h} a_\epsilon(u,v)$, where $\mathcal{T}_h$ is the grid on Ω. Due to additivity of the integral with respect to integration domain, this is trivially fulfilled for the weak forms we consider here.

As a result, retrieving matrix entries from neighbors and adding them where DOFs belong to multiple subdomains correctly constructs an overlapping matrix from nonoverlapping ones. There is one exception however: At the boundary $\partial\Omega_j$ of the overlapping domain, retrieving the entries of an interior DOF will not yield the correct result since the integration domain of said interior DOF extends beyond Ω_j as shown in fig. 4.7.

We circumvent this issue by not sending matrix entries from $\hat{A}_k$ from process k to process j, but instead assemble a new matrix $\hat{A}_k^j$ on the domain $\Omega_j \cap \hat{\Omega}_k$ for that purpose as shown in fig. 4.8. In practice, we communicate the partition of unity Ξ_j generated from the already available matrix graph of A_j to all neighbors, and use $\Omega_j \cap \hat{\Omega}_k = supp(\Xi_j) \cap \hat{\Omega}_k$ as a convenient proxy to the desired domain snippet. Since we have the decomposition

$$\Omega_j = \hat{\Omega}_j \cup \bigcup_{k\in E_j} \Omega_j \cap \hat{\Omega}_k,$$

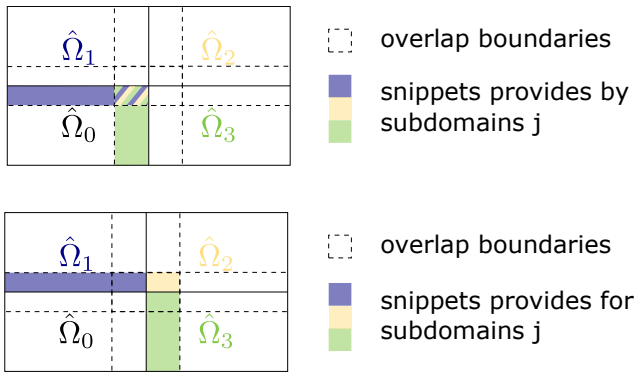


Fig. 4.8: Subdomain snippets $\Omega_j \cap \hat{\Omega}_k$ used for assembly of algebraic overlaps, ensuring correct entries on overlap boundaries.

all snippets together form the desired overlapping subdomain. Exploiting additivity of the bilinear form a and thereby the FE matrix entries, this assembly strategy leaves us with the correct overlapping matrix A_j for Ω_j. Finally, we can embed these components in an extension to our GenEO implementation from [SRS20].

4.4 Online/Offline Approach in Localized Model Order Reduction

Since the GenEO basis is defined per subdomain, some basis vectors may be reused when solving multiple related PDE problems on the same domain. In many realistic applications, a substantial amount of computational effort is spent on solving the GenEO eigenproblems. Therefore calculating the entire basis once in an "Offline" step and only adapting subdomains that changed "online" promises significant cost reductions. Figure 4.9 gives an overview of the procedure detailed in the following.

For the offline step, we perform a coarse space construction as introduced in the preconditioner case (section 4.2.1). For the online step, we assume a problem in weak formulation with bilinear form $\tilde{a}$ and linear form $\tilde{b}$. We further assume that they are largely identical to the original problem, in the sense that both $a_{\Omega_j} = \tilde{a}_{\Omega_j}$ and $b_{\Omega_j} = \tilde{b}_{\Omega_j}$ holds for many $j \in 1, ..., N_\Omega$. We refer to subdomains not fulfilling this condition as "active".

Being able to reuse parts of the GenEO basis in the online step hinges on the fact that the eigenproblems eq. (4.2) defining the coarse basis for subdomain Ω_j are based on the bilinear form a_{Ω_j}. By definition, the eigenproblem therefore only differs in active subdomains, and so do the eigenvectors.

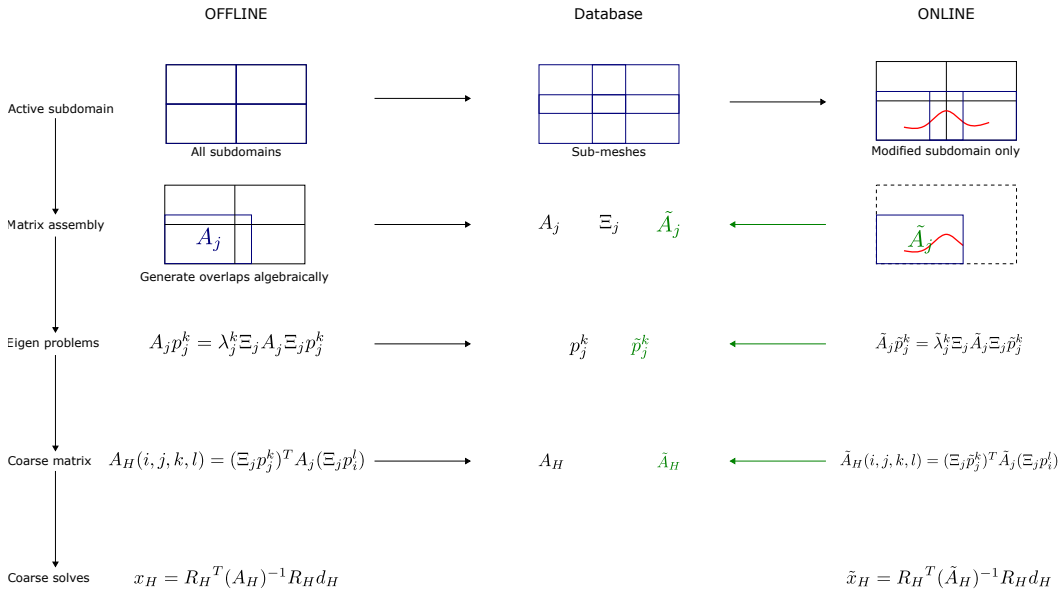


Fig. 4.9: Main steps of the online/offline method reusing the GenEO basis contributions on subdomains where the online PDE problem is equivalent to the previously computed offline one.

The new coarse matrix $\tilde{A}_H$ needs to be updated wherever its entries of the form $(\tilde{A}_H)_{(i,k)(j,l)} = (\Xi_j p_j^l)^T A_j (\Xi_i p_i^k)$ differ from their counterpart in A_H. This is the case when i or j are an active subdomain, and the entry is not trivially zero in case i and j are not overlapping.

This coupling between overlapping subdomains implies that basis functions from subdomains adjacent to active subdomains are needed for the coarse matrix update, even if they are not active themselves. Those basis functions can be drawn from the offline step however.

In the preconditioner setting, we consider this method not very promising, since local solves on all subdomains will be performed regardless (according to the additive Schwarz method) and parallelization would likely not be able to exploit the cost reduction. However, when applying GenEO as an LMOR method, we can expect to reduce cost for an online run from N_Ω eigensolves to one eigensolve per active subdomain, plus the solution of the resulting coarse system.

Note that in fact, the fraction of active subdomains is arbitrary. However, this method is most efficient for few active subdomains out of a large number of subdomains.

In a future application in solving a UQ problem on composite materials using MLMCMC, we will employ this coarse approximation space as a coarse level, while using a GenEO preconditioned Conjugate Gradient (CG) solver as a fine level. We

expect the fine model to scale up to around 10000 processor cores (similar to our application in [Rei+18]). Assuming that the uncertain parameter is the location and shape of a material defect spanning around 10 subdomains, we can expect a speedup factor of around 100 for coarse model evaluations compared to full fine level solves. Due to the online/offline method, only a single processor core can handle a coarse model evaluation. This in turn links up perfectly with the parallelization scheme in chapter 5. Overall, with this approach, we can expect to solve UQ problems on a challenging large-scale model whose simple forward evaluation was out of reach up until a few years ago [But+20b].

5 Modular and Parallel HPC Implementation

As part of this thesis, the multilevel and multiindex methods introduced in chapter 3 were implemented, reviewed and published in the open-source MIT Uncertainty Quantification Library (MUQ) [@Par+] software package. The key innovations are:

- A modular implementation supporting sequential Multilevel Monte Carlo (MLMC), Multilevel Markov Chain Monte Carlo (MLMCMC) and Multiindex Markov Chain Monte Carlo (MIMCMC) methods in a single code base supporting both C++ and Python,
- a massively scalable implementation of the same algorithms supporting distributed High Performance Computing (HPC) environments while retaining a modular architecture,
- and a unified interface for controlling the various degrees of freedom offered by multilevel and multiindex methods.

All these are achieved while carefully embedding the new architecture in the existing modular structure of MUQ. This allows the use of shared and well-tested code, flexibility in extending the new algorithms and easy application of the new algorithms to existing model implementations.

Large parts of this chapter's content will be published as part of a future general publication on MUQ, written together with Matthew Parno and Andrew Davis. This publication includes the description of the general architecture of MUQ as well as sequential MLMCMC/MIMCMC, where the author is mainly responsible for the latter. The part describing the parallel implementation in turn is published in [See+21], which demonstrates its scalability for a simple Poisson case and a large-scale Tsunami model. Here, the author was mainly responsible for the Uncertainty Quantification (UQ) side using the software presented here, while coauthor Anne Reinarz was mainly responsible for the tsunami forward model implementation using the Exascale Hyperbolic PDE Engine (ExaHyPE) framework.

We begin with a brief overview of MUQ's design philosophy in section 5.1. After a review of the existing Markov Chain Monte Carlo (MCMC) components in section 5.2, we proceed to present how the new multilevel and multiindex variants are embedded in that architecture in section 5.3. Section 5.4 presents the architecture of the fully parallelized implementation. A scheduling framework for the parallel code is introduced in section 5.5, followed by a discussion on the maximum scalability the architecture can achieve in section 5.6.

5.1 Introduction to MUQ

The increase in algorithmic complexity when comparing basic MCMC with MLMCMC is representative of a more general development in UQ: Increasingly efficient and powerful mathematical methods imply an increase in algorithmic complexity. For many other advanced methods like surrogate models (e.g. [Dav+21]) or transport maps (e.g. [Mar+16]), the same holds true. As a result, software components gain complexity as well and developing a well-designed and well-tested shared code base becomes increasingly valuable, if not even inevitable.

The goal of MUQ is to provide a powerful, general-purpose framework for UQ. Assumptions on user-supplied models should be strictly minimal in order to keep the framework model-agnostic and compatible with a wide variety of existing model codes. The framework's capabilities should range from HPC-grade applications to easy-to-use rapid prototyping of UQ algorithms. By allowing the reuse of as many components as possible, implementing future methods should require minimal development effort.

These requirements are fulfilled by MUQ in the following way:

- **Model-agnosticity**
  This is achieved by introducing lightweight and abstract model interfaces. For example, for sampling methods it is enough to supply the desired density as part of a `SamplingProblem`. Derivatives may be supplied by the user if an efficient way to compute them exists; otherwise, numerical derivatives will automatically be employed when more advanced UQ methods require it.

  The models themselves can be represented in MUQ's model graphs, which allow easy construction of complex model setups and reuse of model components. They further aid in handling model derivatives, for example by automatically employing chain rules to couple model components' derivatives.

MUQ has been used in conjunction with a number of model codes. Essentially, any C++ or Python compatible code can, with very little adaptation, be used as a model.

- **Modularity**
  The mathematical theory behind UQ as well as the associated algorithms often already expose a certain degree of modularity. For example, in Markov chain Monte Carlo methods, there exist a number of interesting choices for the crucial proposal density (see e.g. chapter 3).

  We mirror these mathematical components in a corresponding software architecture using object oriented programming. On one hand, the architecture that has already proven successful on the theoretical side leads to a well-structured code accessible to mathematicians; on the other hand, implementing new methods building on these theoretical frameworks only necessitates minimal adaptations beyond the existing components. That reuse of components further implies technical advantages like allowing for a well-tested code base and a community familiar with the same tools.

- **Ease of use and HPC**
  Catering towards both HPC and quick method development needs is clearly a challenge. We address this by developing MUQ in C++ with MPI support for large-scale parallelism, while providing extensive Python bindings exposing almost all MUQ functionality. Since the Python bindings correspond directly to the underlying C++ architecture, it is easy to switch between the two and for example port a Python prototype code over to a powerful C++ application.

5.2 Abstract Markov Chain Monte Carlo Framework

Following algorithm 1, MUQ implements MCMC using abstract data structures and objects that each execute a different step in the algorithm. In particular, the main difference between most MCMC type methods are proposal densities and sometimes the kernel. Abstract “proposal” and “kernel” objects reflect this distinction in MUQ. As a result, implementing new methods requires only minimal adaptations in the form of new versions of the algorithm’s building blocks. Similarly, applying various methods to the same problem becomes a matter of simply switching out the desired components.

The first of these components is an `AbstractSamplingProblem` (see fig. 5.1), which at a minimum provides a function that evaluates the logarithm of the target distribution (in a Bayesian setting, up to the proportionality constant). Some MCMC algorithms require more information—for example, likelihood or prior evaluations. Therefore, children of this base class provide interfaces that expose this structure. Optionally, an `AbstractSamplingProblem` can also implement functions that evaluate quantities of interest or derivatives of the target distribution. However, only the log-target is absolutely required. Therefore, just like in the theory, we do not make any assumptions on the underlying type of model and allow direct coupling to arbitrary model codes.

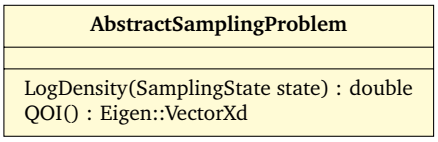


Fig. 5.1: `AbstractSamplingProblem` interface (some methods left out for clarity).

For the actual MCMC method, we have the following components:

- The `SampleCollection` object (see Figure 5.3) stores the actual samples generated in the chain. Furthermore, it offers a number of operations on samples, such as computing the mean, expected values, or other moments.
- The `SingleChainMCMC` class is responsible for stepping through the chain, tracking progress and determining which samples are to be saved. This can, for example, be exchanged for a parallel version that allows running multiple independent chains instead.
- Whether a proposal is accepted as the next step in the chain is something that varies across methods, therefore the abstract `TransitionKernel` provides an interface for that task. For example, the `MHKernel` represents a standard Metropolis Hastings kernel as above, while `MIKernel` is a modified variant for multiindex MCMC.
- Finally, there is a wide variety of proposal strategies, and so `MCMCProposal` provides a suitable abstract interface. For example, `MHProposal` implements standard Metropolis Hastings proposals, whereas `AMProposal.h` provides an adaptive method.

Figure 5.2 shows how that architecture functions for a standard Metropolis Hastings MCMC as introduced above. Note that, at every level components can easily be

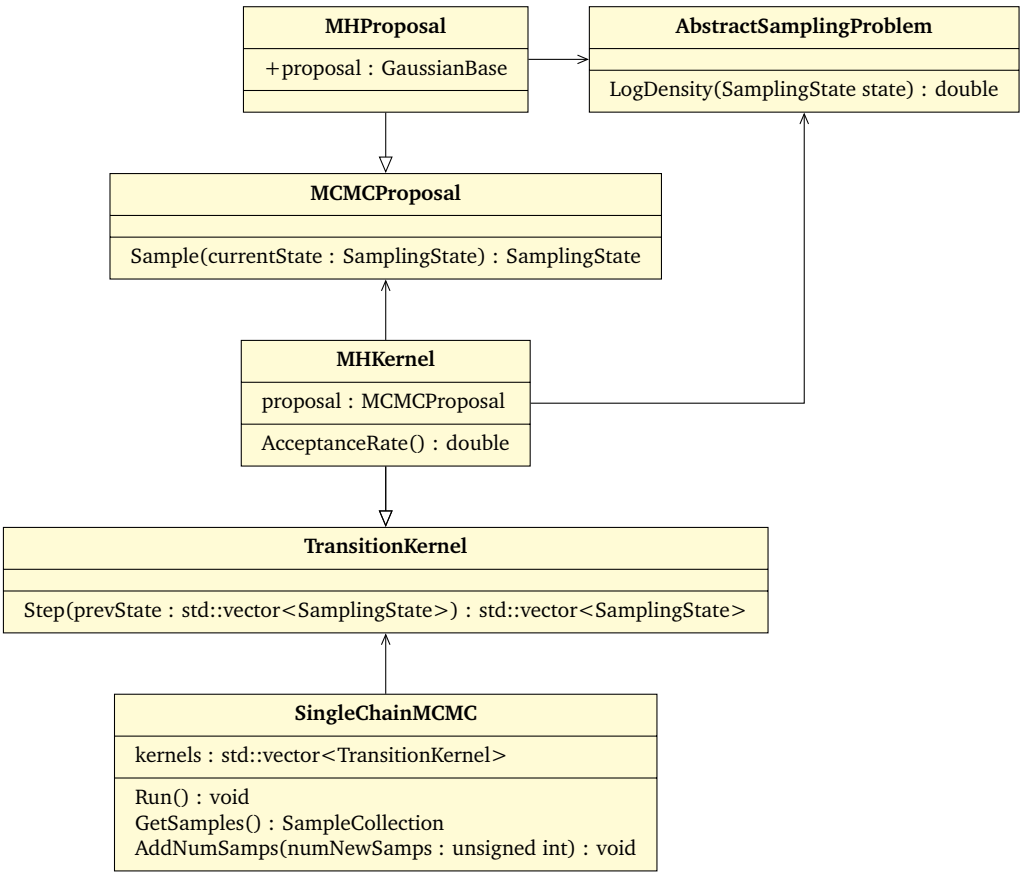


Fig. 5.2: Single chain MCMC architecture for standard Metropolis Hastings MCMC. Note that this depiction is simplified by omitting some class members for clarity of presentation. Simple arrows indicate references between the classes, whereas triangle shaped arrows indicate inheritance.

exchanged. This architecture is capable of easily extending to all MCMC type methods known to us, many of which are readily available in MUQ.

5.3 Multilevel / Multiindex MCMC

5.3.1 Internal architecture

By careful design, we can reuse much of the MCMC architecture introduced above (see Section 5.2) for a multilevel implementation. For clarity of presentation, we do not show the supported more general multiindex case here, and focus on multilevel as its simpler special case. Nevertheless, all components introduced here naturally extend to multiindices and are named accordingly.

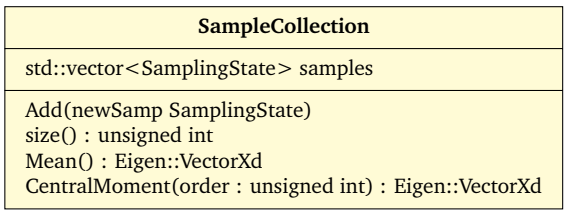


Fig. 5.3: The `SampleCollection` class, which stores samples from a distribution and provides implementations to compute common statistical information such as the mean or other moments.

We begin constructing the architecture already at the telescoping sum. This is of particular importance since each component of the telescoping sum should be estimated from independent samples, and therefore proposals derived from coarser chains may not be shared between them. Further, the number of samples needed for each of these differences usually varies, which makes separating them a cleaner solution.

As depicted in Figure 5.4, we therefore represent each component of the telescoping sum as an instance of the `MIMCMCBox` class, which internally sets up a fine and a coarse chain by means of two `SingleChainMCMC` instances. Each of these chains receive a `MIKernel`, which implements the particular acceptance probability defined in the MLMCMC method 3.3.

The particular kernel of MLMCMC requires both a coarse proposal, usually a sample from a coarser chain, and a fine proposal complementing it in case the finer level expands parameter dimension. For the latter, any existing proposal density may be used; coarse proposals drawn from finer levels are implemented by the `SubsamplingMIProposal`, which itself draws samples from an associated coarser chain.

In order to provide proposals for the finer chain, the `MIMCMCBox` hooks up its associated `SubsamplingMIProposal` to the coarser chain. The coarser chain itself requires coarser proposals again, which recurses until the coarsest level is reached. However, only the aforementioned two chains are used to compute the `MIMCMCBox`'s contribution to the telescoping sum, while the coarser ones only serve to deliver proposals. Therefore, the latter are not publicly exposed and only used internally by the `MIMCMCBox`.

The interplay of objects set up by the `MIMCMCBox` is shown in section 7.1. Note that here we do not show the corner case of level zero (the coarsest one). In that case, no proposals are required from coarser chains and the architecture collapses down

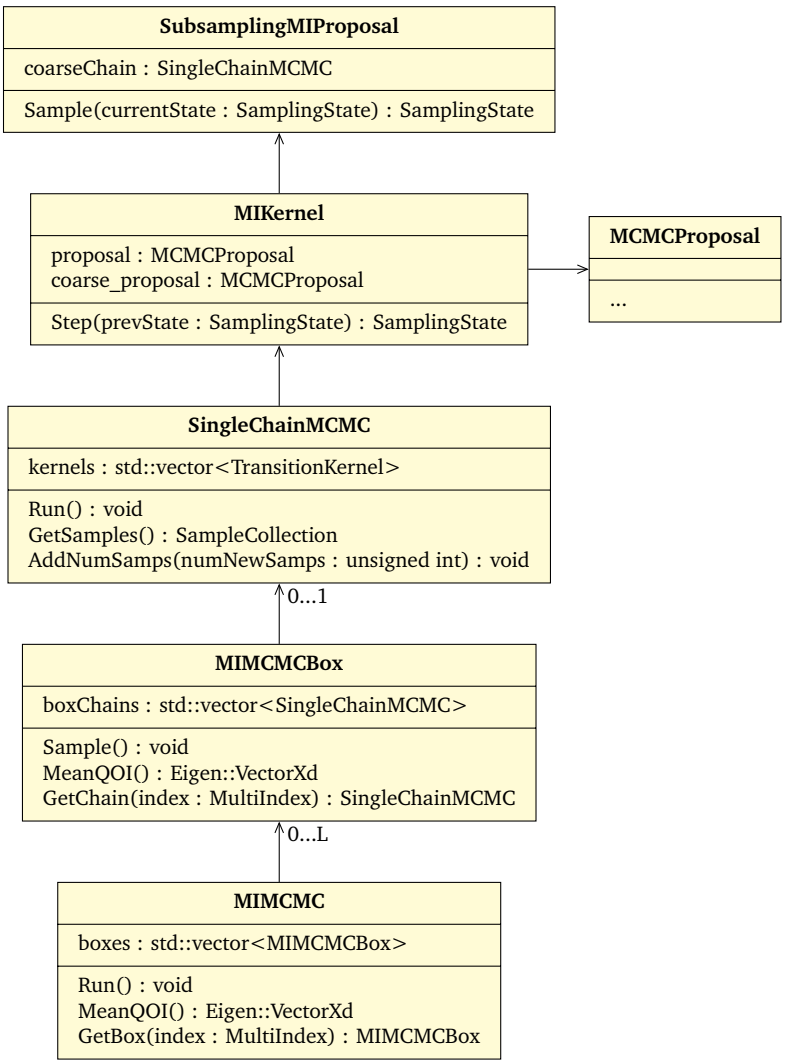


Fig. 5.4: Multilevel MCMC architecture. Note that this depiction is simplified by omitting some class members for clarity of presentation.

to a simple one-level MCMC. Likewise, level 1 only requires a single coarse chain for proposals.

5.3.2 Model interface

For MCMC it is sufficient to specify a kernel, a proposal density and a posterior density to sample from (see Section 2.4.3). A Multilevel MCMC method requires some more information to be specified. Clearly, instead of a single posterior density, we now need an entire hierarchy of posteriors originating from varyingly coarse models. Likewise, proposals can be handled differently across levels.

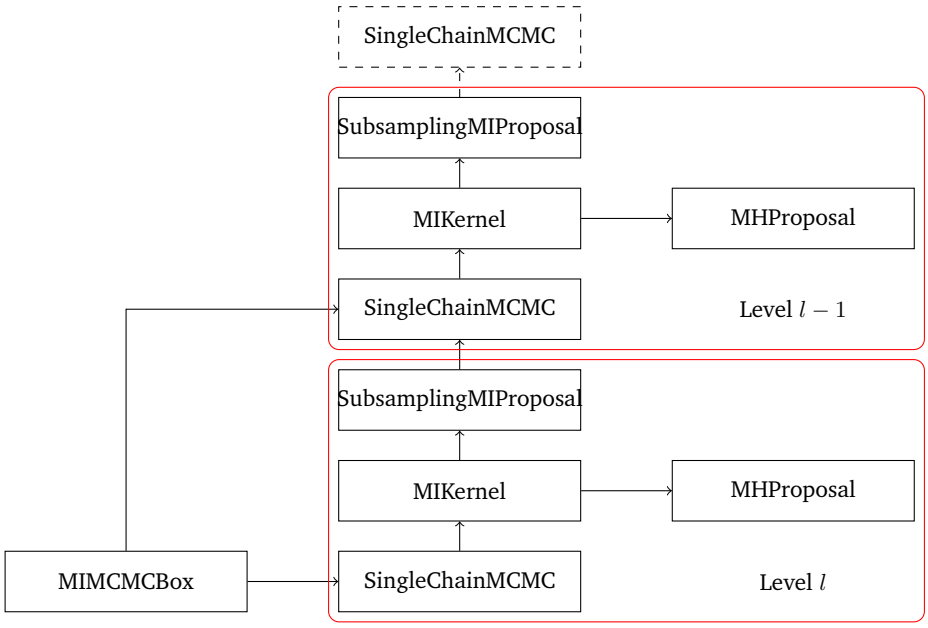


Fig. 5.5: Multilevel MCMC architecture.

In order to provide a unified way of specifying the various components needed for a Multilevel or Multiindex MCMC problem, we introduce a class `MIComponentFactory`, explained in detail below. All of MUQ's MI/MLMCMC type methods expect the user to provide an instance of a `MIComponentFactory` subclass. As a result, the particular method to be used can be swapped out without necessarily touching the problem definition.

MIComponentFactory
FinestIndex() : MultiIndex SamplingProblem(index : MultiIndex) : AbstractSamplingProblem Proposal(index : MultiIndex, samplingProblem: AbstractSamplingProblem) : MCMCProposal CoarseProposal(index : MultiIndex, coarseProblem : AbstractSamplingProblem, coarseChain : SingleChainMCMC) : MCMCProposal Interpolation(index : MultiIndex) : MIInterpolation StartingPoint(index : MultiIndex) : Eigen::VectorXd

Fig. 5.6: `MIComponentFactory` interface.

Its methods (see Figure 5.6) serve the following purposes:

- `FinestIndex` defines the index of the hierarchy's finest model. In the multilevel case, that is simply the finest level L.
- `SamplingProblem` returns the `AbstractSamplingProblem` representing the posterior density for a given model index. For index zero, the coarsest model

should be used, while the finest model should be returned for the value specified in `FinestIndex`.

- `Proposal` defines how proposals are drawn on each level. Any existing or user-defined proposal density, represented by an `MCMCProposal`, may be used here. Note that, while for the coarsest level this is a regular MCMC proposal, on finer levels this is only the fine component of the proposal, relevant to control parameters not present on coarser levels.
- `CoarseProposal` specifies how samples from the next coarser chain are to be drawn as proposals for the given model index. Typically, in order to subsample, this means advancing the coarse chain by a number of steps and only using the last state as a coarse proposal.
- `Interpolation` defines a method to combine fine and coarse proposals from above, by means of an `MIInterpolation` instance. Typically this means simply concatenating coarse parameters and fine ones not yet present on the coarse chain. In more complex cases, this could be an actual interpolation type operation, for example when coupling a finer PDE model to a coarser ODE model. Again this can be specified independently for each level.
- `StartingPoint` finally specifies the chain's initial state. Ideally it is chosen in the mass of the distribution, alternatively a more extensive burn-in may be necessary to avoid a perturbed estimate for small numbers of samples.

5.4 Parallelized Multilevel / Multiindex MCMC

Our new, highly scalable parallel implementation of MLMCMC and MIMCMC (see chapter 3) is presented in this section. While parallelization in classical Monte Carlo (MC) is trivial since samples are independent by definition, MCMC introduces data dependency through proposals depending on the previous step. In the case of MLMCMC, we use coarser chains' samples as proposals, which introduces data dependency between levels. There are, however, multiple opportunities for parallelizing MLMCMC:

- Models: The forward models themselves may be parallelized. In fact, for large models like the tsunami model we introduce in section 6.3, that is even necessary due to memory constraints.

- Chains: Instead of running a single Markov chain, multiple chains can be run in parallel and their samples combined. It is beneficial to not purely rely on this approach though since each chain requires a burn-in phase.
- Levels: Contributions to the multilevel telescoping sum in section 3.3 can be evaluated in parallel.

Exploiting all those clearly introduces significant technical complexity. Therefore we provide our implementation as part of the MUQ C++ library [@Par+]. The main goals of this implementation are:

- Parallelism: All levels of parallelism from above are supported.
- Simple user interface: We hide the intricate details of communication from the user. The algorithm can be tweaked, but defaults suffice to get started.
- Model-agnosticity: The MLMCMC algorithm as detailed in section 3.3 only requires simple forward evaluations of the model. This theoretically allows coupling to arbitrary forward models without introducing e.g. derivatives of the model map. We retain this model agnosticity in the sense that any forward model that can be called (possibly through wrappers) from C++ can be used.
- Modularity: MUQ is, from the ground up, designed as a modular framework. Its modularity is closely modeled after the respective mathematical objects. We extend this concept to our parallel MLMCMC implementation by building on top of MUQ's existing MCMC stack and, as detailed in the following, constructing modular parallel units.

5.4.1 Model interface

The parallel implementation follows the same interface as the sequential version introduced above. The key difference is that here, an additional function `SetComm` is required that allows the user to pass a subcommunicator to the forward model (see fig. 5.7).

This is essential, as it allows MUQ to control the distribution of tasks across processors, and in particular allows segregating groups of processes that compute forward models independently.

Overall, this allows to exploit parallelism on the UQ side with minimal intervention regarding the forward model implementation. For forward models that run sequentially, even the step of passing a subcommunicator to it can be ignored. In that

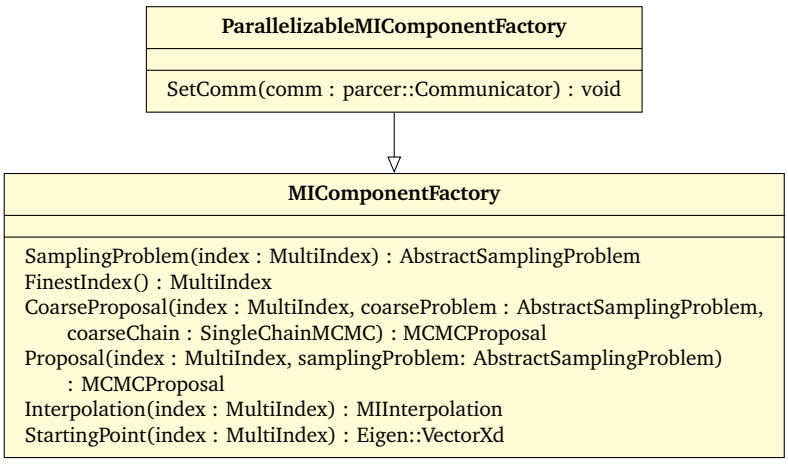


Fig. 5.7: `ParallelizableMIComponentFactory` interface.

case, a model hierarchy already implemented for use with a sequential MLMCMC or MIMCMC method can immediately be used in a parallel setup.

5.4.2 Internal architecture

Our parallel process layout is shown in fig. 5.8. We define the following roles:

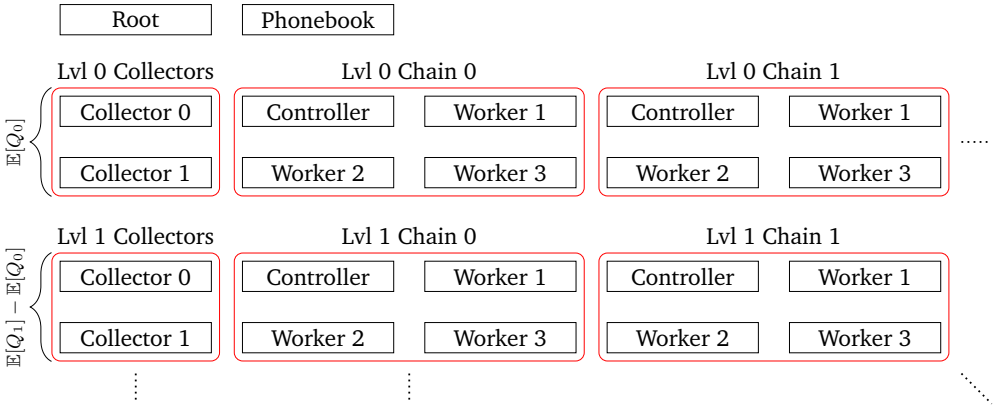


Fig. 5.8: Parallel process layout.

- Fixed roles: These are assigned to specific processes at the start of the parallel method. All other processes wait to be assigned a dynamic role.
  - Root: This process is responsible for launching the parallel method, assigning tasks to other processes and requesting collectors to begin collecting a certain number of MCMC samples. It is also the best place

for users to implement custom (possibly adaptive) sampling strategies, in the sense of how many samples are to be drawn.

  - Phonebook: The phonebook tracks what dynamic roles processes are currently assigned to. Most importantly, it tracks which chains are currently sampling and which ones hold a new sample ready to be picked up by other processes. Further, the phonebook can infer from that the computational load on a given level, since the relation between requested samples and assigned resources is available here. Therefore, it is also the key component in dynamic load balancing across levels and chains.

- Dynamic roles: These roles may be assigned and reassigned at any time. In particular, this permits dynamic load balancing and possibly more advanced sampling strategies.

  Workers and controllers solve forward models, where workers share the load of running a single model evaluation and controllers additionally run the inherently sequential MCMC chains; consequently, they are assigned (and, in case of dynamic scheduling, reassigned) synchronously. This is facilitated by Message Passing Interface (MPI) subcommunicators, which are passed through to and should be used by the user's model.

  - Worker: Workers are responsible for running the user's forward model, specifically the user's implementations of `AbstractSamplingProblem`. Mathematically, they provide parallelized evaluations of the posterior and quantity of interest on a given level for a given parameter. They listen to their respective controller's `ParallelAbstractSamplingProblem` to signal the beginning of a evaluation for a specific parameter θ. Once the signal arrives, they execute the user-implemented `LogDensity` method. Since all workers of a work group are called synchronously, the user can easily supply models assuming lock step parallelism.

  - Controller: Each controller is responsible for a running a multilevel MCMC chain according to algorithm 3. Specifically, a controller on level l contains a chain on level l and one on $l-1$, forming part of a summand of the telescoping sum eq. (3.1) (with the obvious exception of the coarsest level 0). An instance of `ParallelAbstractSamplingProblem` is set up for each model needed. It provides an intermediate layer between instances of the user-implemented `AbstractSamplingProblem` instances running on multiple workers and the controller, allowing to transparently distribute model executions. As a result, neither the user nor

the inherently sequential MLMCMC chains need to concern themselves with synchronizing worker processes to run forward models in lock step.

The chains themselves are implemented using existing MUQ components: They are `SingleChainMCMC` instances with `MCMCKernel` implementations matching the acceptance probability of algorithm 3. Drawing samples from coarser chains as proposals is implemented with an `MCMCProposal` requesting coarser samples from other controllers via the phonebook process.

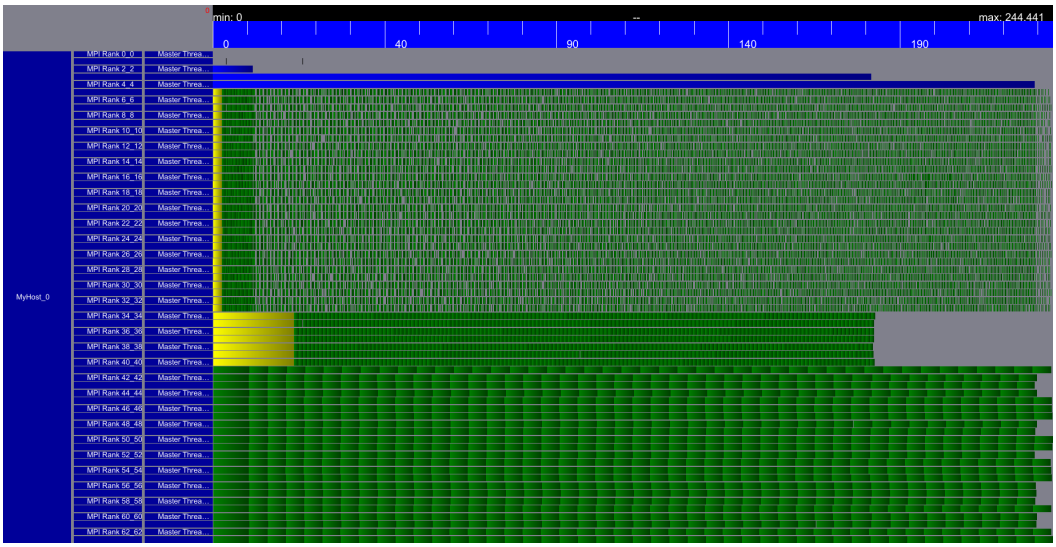
– Collector: Collectors request samples from controllers via phonebook in order to form a summand of the telescoping sum eq. (3.1). Multiple collectors may be responsible for a single level. Together, they hold a `DistributedCollection`, an existing class in MUQ for storing and computing statistics on samples in a parallel system.

Note that this architecture is defined on process level (more specifically, in terms of MPI ranks). Thread-level parallelism can easily be exploited by worker processes. In fact we make use of this through Intel Threading Building Blocks (TBB) in our ExaHyPE Tsunami model, since ExaHyPE exhibits better performance characteristics with few MPI ranks per node each making use of several threads.

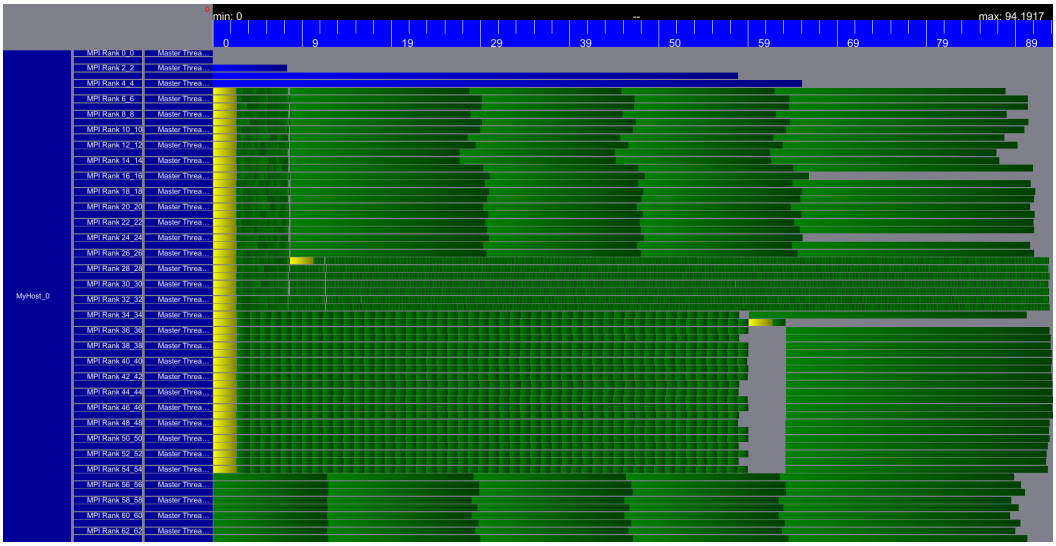
In order to make the parallel architecture as modular as sequential MUQ code, each of the above roles provide an MPI interface based on requests mimicking function calls. This allows recombining the parallel components in analogy to object orientation in order to implement other algorithms as well. For example, work groups as introduced above are based on a `ParallelAbstractSamplingProblem`, which in turn can be used to easily employ any sequential sampling algorithm on a parallelized model. Likewise, the phonebook could be swapped out for an alternative implementation with identical MPI interface, allowing for alternative load balancing strategies.

5.5 Dynamic load balancing

Data dependencies in MLMCMC (see algorithm 3) introduce a load balancing problem, since coarser chains need to provide proposals to finer ones only until the desired number of fine samples is computed. Estimating the ideal distribution of computational resources across levels is far from trivial or outright impossible in



(a) An exemplary run without load balancing. The top rows contain processes delivering level 0 samples to serve as proposals as well as forming part of the telescoping sum. After a short time the latter is completed, while occasionally proposals are needed. As a result they are frequently idle.



(b) An exemplary run with load balancing. As soon as the aforementioned processes run idle, they are reassigned to finer levels and we achieve near full machine utilization.

Fig. 5.9: Dynamic load balancing in parallel MLMCMC. Run time is on the horizontal axis, while process indices are on the vertical. Green boxes indicate model evaluations, while yellow boxes indicate chains' burnin phases. In these examples, model evaluations clearly differ strongly in run time.

realistic applications, especially when adaptively determining the number of samples per level.

Therefore, our parallel MLMCMC implementation provides a load balancing mechanism to reassign worker processes to different tasks once samples on another level are more critical to runtime. Figure 5.9 illustrates this load balancing mechanism for a small test run.

Load balancing is implemented as part of the phonebook rank, since it keeps track of how samples are passed around. Levels with low load are detected when samples on that level are provided but not quickly picked up, while a high load is in turn detected when sample requests remain queued. Unanswered sample requests originating from other chains are given a higher impact than requests originating from collector processes, since the first case implies chains waiting and therefore bad machine utilization.

Models may have strongly varying run times. A new group of processes assigned to a certain level only reduces that level's load once it actually provides its first sample. This implies the danger of reassigning tasks too frequently or too infrequently. In order to stabilize the load balancer, the respective model run times are inferred by the phonebook process by the frequency of samples provided. Based on that, scheduling will only take place at the time scale of the respective model evaluations.

Note that this load balancer is unaware of the specific types of proposals or MCMC kernels being executed. As a result, it can, for example, also be applied in the MLMC setting as well. Also, all model specific tuning parameters are determined dynamically, so no user intervention is required.

5.6 Limits of Parallel Scalability

All roles in the parallelization strategy above (see fig. 5.8) are trivially scalable; the only component not able to scale arbitrarily is the phonebook, as for now it is restricted to a single rank.

However, as it turns out, the phonebook's load (in the sense of fraction of run-time not spent waiting for new incoming MPI messages) is very low in the scalability experiments of section 6.1. The exact results are shown in fig. 5.10. Extrapolating from those, a parallel run with around 10^4 should leave the phonebook at around 10% load. A full load of 100% would be reached at around 10^5 ranks. High loads on

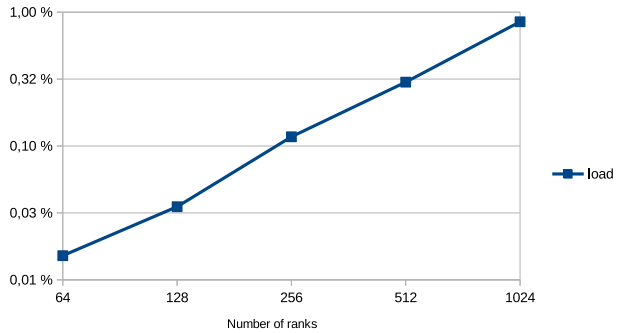


Fig. 5.10: Load (i.e. fraction of non-idle time) of the phonebook rank in the application from section 6.1.

the phonebook should however be avoided, since it could lead to blocking valuable worker processes.

This estimate comes with a caveat: The distribution of load as well as overall time scales directly depend on the forward model employed; for example, a model hierarchy taking twice the amount of time for a forward solve immediately halves load on the communication infrastructure, and vice versa. On the other hand, the numbers above stem from a model that intentionally only requires one processor core for each model level in order to produce a high load on MPI communication. The parallel architecture however supports parallelization across forward models themselves, which has no effect at all on the amount of communication needed for our parallel method. So, in an otherwise equivalent setup (especially identical run times of forward models) but with a forward model parallelized across 100 cores, the above example can easily scale to 10^6 cores at around 10% load on the phonebook.

In an application where further scalability beyond the current architecture is required, two main approaches should be relatively easy to implement.

- Multiple instances: The above architecture may exist in several independent instances. This requires little to no change to the implementation. It does however come with the caveat that information exchange between instances is practically impossible, and in particular load balancing across instances would be unavailable. Depending on the application, load balancing within each instance may however be sufficient.
- Multiple phonebooks: Running multiple phonebooks is somewhat more invasive, but the existing implementation is modular enough to keep changes minimal. When having multiple phonebook instances responsible for a limited

set of processes each, load balancing would be restricted to take place within these sets. However, in contrast to the previous approach, a single set of collector ranks would exist, allowing much finer control over the algorithm from a single root rank.

Since the scalability limit was not encountered in the applications detailed here, these extensions are left to future work.

6 Applications

Here we show numerical applications of the algorithms and software infrastructure introduced above. The main focus is the investigation of the hierarchical Uncertainty Quantification (UQ) methods' behavior in realistic large-scale scenarios as well as the demonstration of parallel scalability for High Performance Computing (HPC) purposes. Large parts of this chapter have been accepted for publication in the Supercomputing 21 conference [See+21]; the tsunami application in section 6.3 is joint work with Anne Reinarz and Leonhard Rannabauer, who were mainly responsible for the forward model.

6.1 Parameter Field Estimation - Poisson Equation

This application is intended to demonstrate the scalability of our new parallel implementation (see chapter 5) of the Multilevel Markov Chain Monte Carlo (MLMCMC) method (see section 3.3). Since it is a widespread example, we choose parameter field estimation in the Poisson equation as our example. Further, since the model itself can be constructed in a computationally cheap way, it lends itself to investigating massive parallel scalability of UQ algorithm and parallelization strategies.

6.1.1 The Physical Model

In this example, our forward model maps a parameter θ that models the uncertainty in the diffusion coefficient to the solution of the Poisson Partial Differential Equation (PDE) evaluated at certain points. The inverse problem consists in estimating the underlying parameter θ from given synthetic data while taking uncertainty in that data into account.

Specifically, we solve the PDE

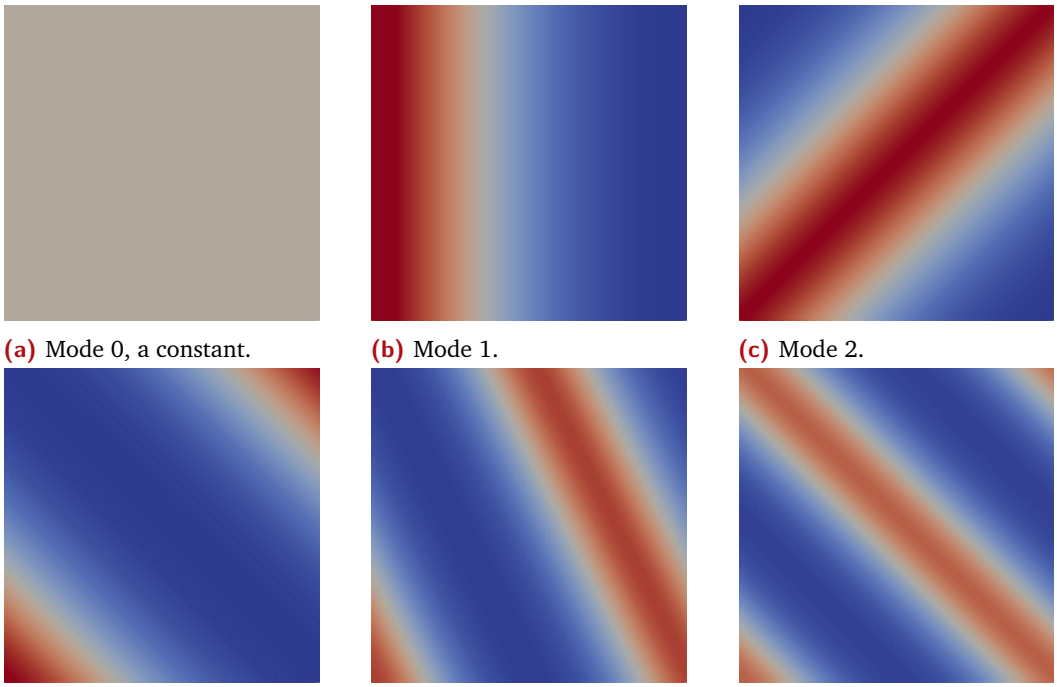
$$\nabla \cdot (\kappa(x, \theta) \nabla u(x, \theta)) = 0 \qquad \text{for} \quad x \in \Omega \text{ and } \theta \in \Theta,$$

where we choose the domain $\Omega := [0,1]^2$ and $\Theta := \mathbb{R}^m$. As boundary conditions, we apply $u(x) = 0$ on the left, $u(x) = 1$ on the right and natural Neumann boundary conditions at the remainder of the boundary. We model $log(\kappa)$, the logarithm of the diffusion coefficient, as a zero mean random field with Gaussian autocorrelation, correlation length 0.15 and variance 1. In order to arrive at a finite dimensional representation of the field, we use an approximation to a Karhunen Loève (KL) expansion which we truncate after m terms, i.e.

$$\log\left(\kappa(x,\theta)\right) \approx \sum_{k=1}^{m} \phi_k(x)\theta_k.$$

In the expansion, $\phi_1, ..., \phi_m$ are the KL modes of largest wave length. Consequently, θ is a vector of KL coefficients. We implement the model in the Distributed and Unified Numerics Environment (DUNE) framework [BHM10] with a $\mathcal{Q}_1$ Finite Element discretization on simple structured grids. To form a three-level model hierarchy for our MLMCMC method, we choose mesh widths of $\frac{1}{16}, \frac{1}{64}$ and $\frac{1}{256}$. Across all three levels, we choose an identical parameter dimension $m = 113$; this specific number carries not much significance beyond making sure we select modes of up to a certain specific wavelength, as detailed later. We use the `dune-randomfield` module to generate the fields efficiently. It is based on circulant embedding [DN97], and we modify the module slightly in order to fit our needs. In particular, we allow injecting specific parameters chosen by Markov Chain Monte Carlo (MCMC) into the generator instead of just drawing independent random fields.

First of all, since we are working in a 2D domain, the truncation of modes is not immediately obvious. It is generally desirable to include low-frequency modes first, since they allow to recover large-scale features of the parameter field we seek in the inverse problem, and we would prefer to capture those over higher-frequency components that likely have only local effects on the physical model. Therefore we place the first index in matrix entry $(0,0)$ of the frequency domain matrix employed in circulant embedding, corresponding to the constant mode. We then proceed to add further modes by forming layers of entries around that (see section 6.1.1). In order to give an idea of the resulting ordering of modes, some representative modes corresponding to entries in frequency domain are shown in fig. 6.1. The generation of random fields is an extensive research field of its own and not in the scope of this thesis, so we refer to literature for further detail on circulant embedding for random fields [DN97].



(a) Mode 0, a constant. **(b)** Mode 1. **(c)** Mode 2.

(d) Mode 4. **(e)** Mode 7. **(f)** Mode 12.

Fig. 6.1: Individual modes of the permeability field used in these experiments.

θ_3	θ_2	θ_6				θ_7	θ_4
θ_{10}	θ_9	θ_8				θ_{12}	θ_{11}
θ_{10}	θ_{11}	θ_{12}				θ_8	θ_9
θ_3	θ_4	θ_7				θ_6	θ_2
θ_0	θ_1	θ_5				θ_5	θ_1

Fig. 6.2: Choice of modes in frequency domain of circulant embedding method. The ordering ensures that modes are unique and the parameter vector θ is ordered by increasing frequency of modes.

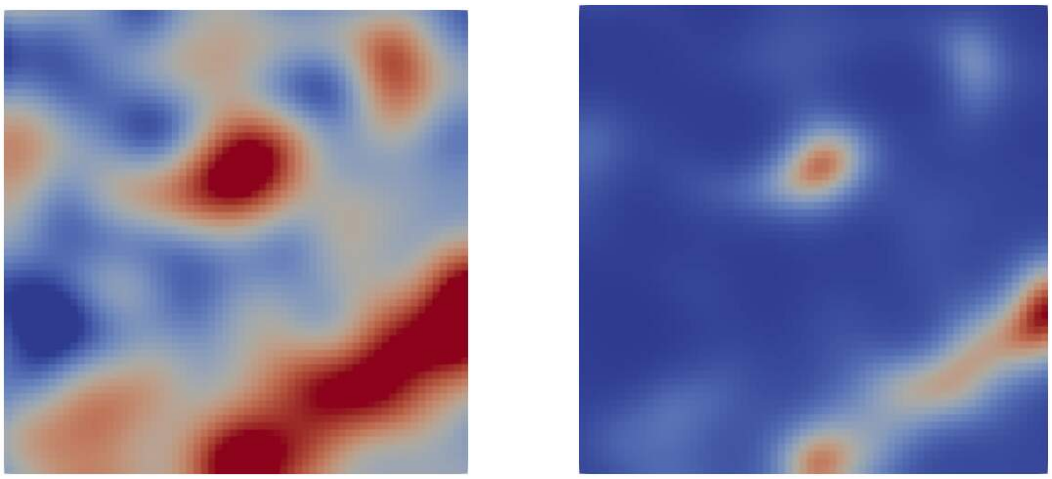


Fig. 6.3: Random field realization $log(\kappa(\cdot, \hat{\theta}))$ and parameter field $\kappa(\cdot, \hat{\theta})$ from random field realization used for synthetic data.

In order to form a Bayesian inverse problem, we generate synthetic data based on a random field $\kappa(x, \hat{\theta})$, where $\hat{\theta}$ is a fixed sample drawn from $\mathcal{N}(0, I)$ (shown in fig. 6.3).

The actual vector of measurements y is then defined by solving the above Poisson problem for $\theta = \hat{\theta}$, evaluating the solution u at a grid of points: $\{\frac{2}{32}, \frac{7}{32}, \frac{13}{32}, \frac{19}{32}, \frac{25}{32}, \frac{3}{32}\}^2$. Based on that, we define our likelihood $\mathcal{L}(y|\theta)$ to be a Gaussian $\mathcal{N}(F(\theta), \sigma_F^2 I)$ with $\sigma_F = 0.01$. Complementing it with a Gaussian prior $\pi_0(\theta) = \mathcal{N}(0, 4I)$, we complete our Bayesian inverse problem.

Note that, by generating synthetic measurements directly from our forward model, we commit an 'inverse crime' [CK19, p. 179]. In realistic applications a model error is inevitable, making it significantly harder to recover the underlying parameters from data accurately. However, for this problem we intentionally accept this simplification: Our focus in this case is algorithmic scalability and not a fully realistic setting. Further, verifying the correctness of the UQ method is somewhat easier without a model error.

As our Quantity of Interest (QOI), we define $Q_k(\theta) = \kappa(x_k, \theta)$ where the x_k form a grid of width $\frac{1}{32}$. This captures the parameter field we seek and, as necessitated by the telescoping sum in eq. (3.1), allows for a consistent dimension in QOI even when varying the parameter dimension across levels.

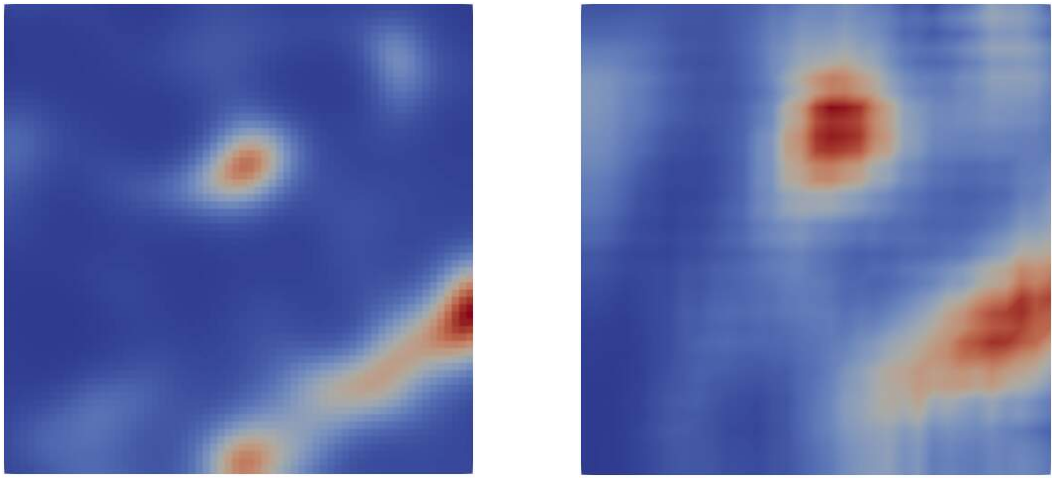


Fig. 6.4: Synthetic "true" field (left) and expected value of multilevel estimator (right).

level l	h_l	DOFs	t_l[ms]	ρ_l	τ_l	$\mathbb{V}[Q_0]$ or $\mathbb{V}[Q_l - Q_{l-1}]$
0	$\frac{1}{16}$	289	3.35	206	137.3	1.501×10^{-1}
1	$\frac{1}{64}$	4225	45.64	17	11.2	1.121×10^{-3}
2	$\frac{1}{256}$	66 049	931.81	0	1.05	4.165×10^{-5}

Tab. 6.1: Multilevel properties of Poisson application. For each level l of mesh width h_l with associated degrees of freedom (DOFs), we show the computational cost t_l and chosen subsampling rate ρ_l. Due to high dimension of QOI in this setting, we only show integrated autocorrelation time τ_l and variance for an single representative component of Q.

6.1.2 Results

In order to fully specify the MLMCMC algorithm for the given problem, it is enough to set a Gaussian proposal on the coarsest level. We choose $\mathcal{N}(0, 3I)$ in order to roughly match the prior. Since we have identical parameter dimensions across levels, no fine level proposals are needed.

The MLMCMC method run with 10^4, 10^3 and 10^2 samples on levels 0, 1 and 2 exhibits properties detailed in in table 6.1 and captures the main features of the parameter field underlying our synthetic data (see fig. 6.4). Clearly some higher frequency detail is not recovered. This, however, is expected due to the limited number of KL modes we include in our parameter space. Note that we can only recover this up to a scaling factor, since the solution is only determined by the parameter field up to a factor. Here, the choice of prior essentially determines the scaling of the solution we observe.

For a more detailed analysis of Bayesian inverse problems based on Poisson equation in MLMCMC, we refer to the original MLMCMC publication [Dod+15; Dod+19b].

In order to investigate parallel scalability of our MLMCMC implementation, we conduct weak and strong scaling experiments on the BwForCluster MLS&WISO Production HPC system. The partition we used consists of nodes with two 16-core Intel Xeon E5-2630v3 CPUs and 64 gigabytes of memory.

As forward model, we use the Poisson model, since its low computational demand allows us to stress the parallelized MLMCMC algorithm itself by running a large number of chains and samples. We use the same inverse problem detailed above, even though the particular inverse problem does not affect the algorithm's communication patterns and therefore parallel scalability.

For the strong scaling setup, we draw 10^4, 10^3 and 10^2 samples on levels 0, 1 and 2 respectively. We further set subsampling rates according to table 6.1, and enable dynamic load balancing. As the timing results in fig. 6.5 show, we achieve linear speedup until relatively large burnin phases and suboptimal load balancing due to few samples per chain occur.

Technically the observed speedup even slightly exceeds linear. That is simply due to the fact that a fixed number of the processes is reserved for book keeping tasks (i.e. the root, phonebook and collector processes). As a result, for increased number of processes, a larger fraction contributes to parallel speedup by generating samples.

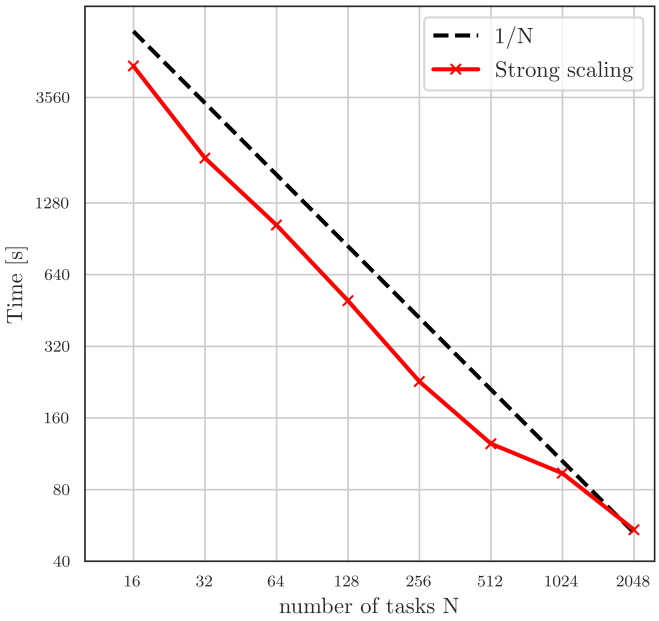


Fig. 6.5: Scalability of the Poisson model problem for 10^4, 10^3 and 10^2 samples on levels 0, 1 and 2 respectively. Subsampling rates etc. are chosen according to table 6.1. The problem setup remains constant as the number of processors is increased.

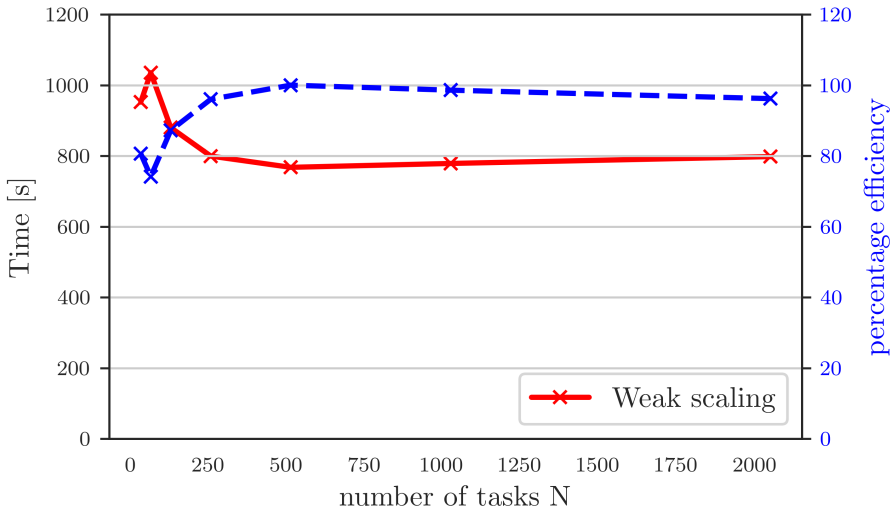


Fig. 6.6: Weak scalability and parallel efficiency of the Poisson model problem. At 64 cores 10^4, 10^3 and 10^2 samples are computed on levels 0, 1 and 2 respectively. The number of samples is modified linearly with the number of processors.

In our weak scaling test, we begin with the same setup as in the strong scaling setting. In particular, we again choose to compute 10^4, 10^3 and 10^2 samples on levels 0, 1 and 2, and solve this problem using 64 processes. We then expand to a range from 32 to 1024 processes while scaling the number of samples on each level linearly in accordance with the number of processes.

The parallel efficiency (given in blue in Figure 6.6) is measured here as $\frac{t_{\text{ref}}}{t_N} \cdot 100\%$, where t_{ref} is the quickest time taken over all runs and t_N is the time taken on N ranks. The initial increase to over 100% efficiency is due to the overhead of phonebook and collector ranks. We achieve fairly consistent results of up to 80 seconds of total run time except for the largest run. The latter is a very extreme scenario though, since the extremely short run time of the coarsest model leads to a significant load on the communication infrastructure. We therefore consider it reasonable to assume that exceeding the ideal range should only occur for significantly larger numbers of processes in more realistic applications.

6.2 Multiindex Markov Chain Monte Carlo - Consistency check

In order to verify that the Multiindex Markov Chain Monte Carlo (MIMCMC) method from section 3.4 was implemented correctly in chapter 5, we show a brief consistency check here. Note that, in order to facilitate testing, this setup is not realistic in certain regards, for example the choice of number of samples or artificial data.

We mainly recreate the multilevel setup in section 6.1, but adapt the forward model to fit the multiindex case. For brevity, only the differences in the inverse problem compared to the multilevel case are described in this section.

6.2.1 The Physical Model

The multiindex model hierarchy is obtained by replacing the discretization of the domain $\Omega = [0, 1]^2$ by a structured grid where mesh width in x and y direction may be chosen independently. So, due to employing a fully structured grid, each element is now a rectangle, and not necessarily a square as in section 6.1.

Each two-dimensional multiindex $\alpha \in \{0, 1, 2\}^2$ then corresponds to a grid of mesh width $(h_{\alpha_0}, h_{\alpha_1})$ where $h_0 = \frac{1}{8}$, $h_1 = \frac{1}{32}$ and $h_2 = \frac{1}{128}$ are chosen. We therefore have

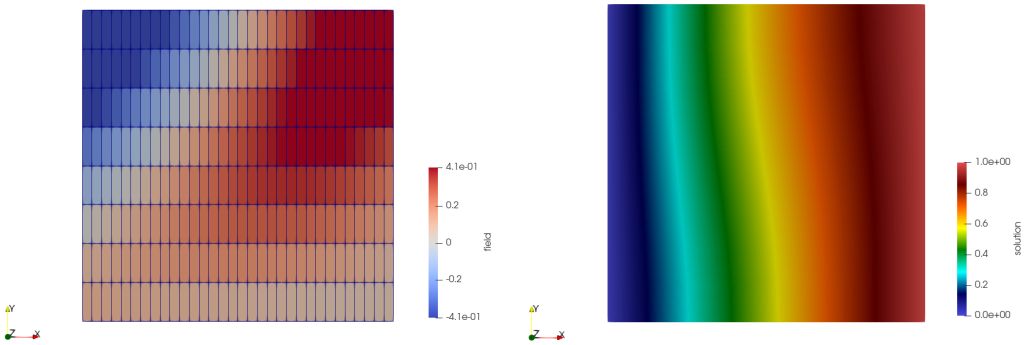


Fig. 6.7: Parameter field κ (left) and resulting PDE solution (right) for a representative choice of θ. The grid belongs to $\alpha = (2, 1)$ and is therefore more refined in x direction.

a total of nine models F_α, and the three "diagonal" indices α with $\alpha_0 = \alpha_1$ in fact form a hierarchy of grids of squares analogous to the multilevel case from section 6.1 again.

On each grid, we solve the same Poisson PDE as in section 6.1.1, again with a $\mathcal{Q}_1$ Finite element (FE) basis and a parameter field κ defined by KL models. In contrast to before, however, we only take the first four KL modes, and therefore $\theta \in \mathbb{R}^4$. A low-dimensional parameter requires few samples for an accurate result, and we can therefore check the accuracy of the output at relatively low computational cost.

The inverse problem is defined in the same way as well, with slightly different variances and choice of QOI and measurements. As prior, we set $\mathcal{N}(0, 5I)$. For the likelihood's, the fidelity variance is chosen as $\sigma_F^2 = 10^{-2}$, which ensures a relatively low posterior variance for the purpose of easily checking expected values produced by the algorithm. Note that, for simplicity, the same priors and likelihoods functions are chosen for all indices, and only the forward models F_α differ.

Model output $F_\alpha(\theta)$ is defined on a grid of points, specifically evaluations of the solution u at points $\{0.1, 0.3, 0.5, 0.7, 0.9\}^2$. In order to complete the definition of the posterior, synthetic data is generated by $F_{(2,2)}(\hat{\theta})$, where $\hat{\theta} := (0, 0, 0, 1)\top$. As before, this choice is an "inverse crime" and not representative of realistic applications. However, it is very helpful for the purpose of checking correctness of the implementation.

Finally, the quantity of interest is defined as

$$Q_\alpha(\theta) = \int_{[0.4,0,6]^2} u_\alpha(x, \theta) dx,$$

$\mathbb{E}_{\nu^{(2,2)}}[Q_{(2,2)}]$	ESS
1.949E-02	46.72

Tab. 6.2: Results of single level MCMC reference run.

2	-4.997E-04	-6.662E-05	-2.065E-05
1	2.245E-03	4.055E-04	-7.076E-05
0	1.583E-02	2.392E-03	-5.496E-04
α_1 / α_0	0	1	2

Tab. 6.3: Resulting estimates of $\Delta_i \mathbb{E}_{\nu^\alpha}[Q_\alpha]$.

where $u_\alpha(x, \theta)$ is the solution of the PDE as defined above for model α and parameter θ. In practice, Q_α is evaluated by numerical integration on the same grid the PDE is solved on.

6.2.2 Results

On the finest model, i.e. $\alpha = (2, 2)$, we run an MCMC chain in order to obtain a reference result. The MCMC method is outfitted with an Adaptive Metropolis [HS98; HST01] proposal, initial variance of 1.0, and variance updates every 100 steps until 1000 samples have been drawn. The chain is started at $\theta^0 = 0$ and is given a burnin phase of 100 samples.

Next, the MIMCMC method is applied to the multiindex model hierarchy above. The MIMCMC method is specified with the same proposal as in the reference run for its coarsest chain. We keep parameter dimension constant across levels, i.e. $R_\alpha = 2$, and therefore do not have fine proposals in this application.

Subsampling rates for proposals from level α are defined as $\rho_\alpha := 2^{\sum_i |2-\alpha_i|}$, and therefore range from 1 on the finest to 16 on the coarsest index. This choice ensures that in the standard proposal routing strategy (see fig. 3.5), subsampling is always increased by a factor of 2 when drawing samples from the next coarser chain. Finally, we request 1000 samples to be drawn for each index, again with a burnin of 100 samples.

From this MIMCMC run, we observe that the magnitudes of components of the telescoping sum (listed in table 6.3) diminishes for increasingly high multiindices. This is exactly as expected and analogous to the multilevel case.

2	1.555E-10	2.404E-09	1.055E-09
1	1.226E-09	2.333E-09	1.468E-09
0	1.289E-08	1.396E-09	4.201E-10
α_1 / α_0	0	1	2

Tab. 6.4: Estimated variance of the estimator $\mathbb{V}_{\nu^\alpha}[\Delta_i Q_\alpha]/N^\alpha_{eff}$, i.e. sample variance over estimated Effective Sample Size (ESS) denoted by N^α_{eff}. See table 6.6 for estimates of N^α_{eff}.

2	1.757E-02	2.031E-02	1.967E-02
1	1.807E-02	2.087E-02	2.025E-02
0	1.583E-02	1.822E-02	1.767E-02
α_1 / α_0	0	1	2

Tab. 6.5: Resulting estimates of partial telescoping sums $\sum_{\alpha' \leq \alpha} \Delta\mathbb{E}_{\nu^{\alpha'}}[Q_{\alpha'}]$. The $\leq$ relation between multiindices is understood component-wise. Therefore, the partial telescoping sum for $\alpha = (0,0)$ consists of only the coarse chain estimate, while the one for $\alpha = (2,2)$ is the full telescoping sum from eq. (3.8).

2	1000	583.632	694.756
1	647.418	586.005	665.159
0	62.9783	575.853	734.519
α_1 / α_0	0	1	2

Tab. 6.6: Estimation of ESS N^α_{eff} for each estimator of $\Delta\mathbb{E}_{\nu^\alpha}[Q_\alpha]$.

When computing the telescoping sum (eq. (3.8)) or parts thereof, we observe in table 6.5 that the overall telescoping sum gives the best approximation of the reference value computed above in table 6.2.

Unfortunately the model hierarchy chosen here does not clearly exhibit variance reduction across levels (see table 6.4). In particular, the gains in more interesting truncations of the telescoping sum like a convex shape, as discussed for Multiindex Monte Carlo (MIMC) in [Haj+16], do not become apparent here. Showing that would likely require either substantial computational effort due to larger model hierarchies or an even cheaper to compute forward model.

Regarding the new strategy for handling proposals in MIMCMC introduced in section 3.4, we observe the benefit in terms of ESS estimates in table 6.6. While the coarse chain ESS is in line with the single level run above (see table 6.2), we observe significantly higher ESS for finer indices, indicating well-informed proposals from coarser chains. Again, this is analogous to the MLMCMC case.

An MIMCMC application to a real-world problem is not part of this work due to time constraints. However, from this brief investigation we conclude that algorithm and implementation deliver plausible results. Since the implementation is identical to the MLMCMC cases presented here, we expect it to offer the same real-world performance demonstrated for MLMCMC. In addition, gains in computational efficiency shown for MIMC in [Haj+16] can be expected and will be investigated in future applications.

6.3 Tsunami Origin Estimation - Shallow Water Equation

This section presents an application of the parallelized MLMCMC to a realistic problem modelling the Tohoku tsunami.

6.3.1 The Physical Model

Tsunami propagation is typically modeled by some variant of the shallow water equations [Beh+10; LGB11, e.g.], allowing for a simulation in only two dimensions. The shallow water equations are obtained via depth-averaging of quantities (esp. momentum) from the more complicated three-dimensional Navier-Stokes equations,

based on the modeling assumption that horizontal length scales are considerably greater than the vertical length scales.

In this setting, we concentrate on the basic shallow water equations with bathymetry source terms (neglecting friction terms or more advanced models for with non-hydrostatic corrections). The resulting equations can be written in first-order hyperbolic form as

$$\frac{\partial}{\partial t}\begin{pmatrix} h \\ hu \\ hv \\ b \end{pmatrix} + \nabla \cdot \begin{pmatrix} hu & hv \\ hu^2 & huv \\ huv & hv^2 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ hg\,\partial_x(b+h) \\ hg\,\partial_y(b+h) \\ 0 \end{pmatrix} = 0, \tag{6.1}$$

where h denotes the height of the water column, (u, v) the horizontal flow velocity, g gravity and b denotes the bathymetry. This hyperbolic system of equations is supplemented by a set of suitable initial and boundary values.

We discretize with an Arbitrary-high-order-DERivative Discontinuous Galerkin Method (ADER-DG) method as proposed in [Dum+08]. It is essentially a predictor-corrector scheme. A high-order solution is found element-locally and then corrected to take into account neighbors by solving Riemann problems along element interfaces. To resolve known high-order issues such as the Gibbs phenomenon a corresponding a-posteriori finite volume sub-cell limiter is applied [DL16]. This limiter detects and revokes problematic ADER-DG solution candidates and recomputes them with a robust Finite-Volume scheme (cf. [Ran+18; RDB18] for details), following the approach by LeVeque et al. [LGB11]. At coastlines the schemes relies entirely on the Finite-Volume limiter, to correctly treat inundation.

As a large-scale example, we invert data from the Tohoku tsunami, which occurred subsequent to an earthquake in the Japan trench in 2011. We assume that the only significant sources of the tsunami are the displacements of the sea floor. In order to initialize the tsunami, we can impose the displacements as an instantaneous deformation of the bathymetry in the resting-lake case – compare respective modeling approaches by Saito et al. [SF09] or Madden et al. [Mad+21]. By keeping the water column constant, the change of the bathymetry is directly translated to the sea surface and generates the tsunami. Gravity, as the main acting force, initiates the propagation of the wave. The tsunami then evolves as a gravity wave. As reference solution for the initial displacements of the ocean floor we use respective simulation results provided by Galvez et al. [Gal+14]. The bathymetry data has been obtained from GEBCO [1].

[1] `https://www.gebco.net/`

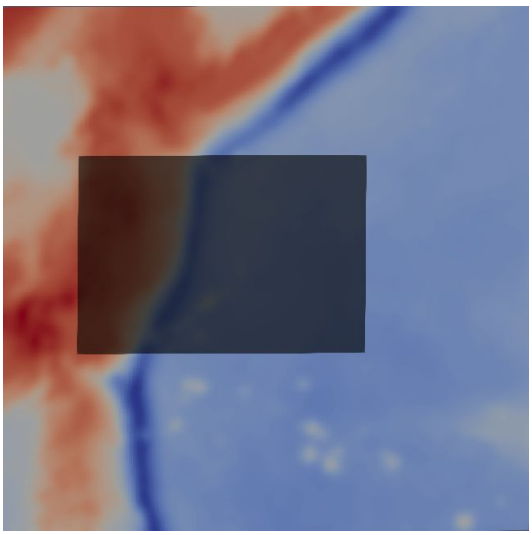


Fig. 6.8: Bathymetry for the full computational domain with darker rectangle denoting parameter values in the prior.

μ	Σ		
	l=0	l=1	l=2
1.85232	0.15	0.1	0.1
0.6368	0.15	0.1	0.1
30.23	2.5	1.5	0.75
87.98	2.5	1.5	0.75

Tab. 6.7: Mean μ and covariance Σ for all three levels.

Our goal is to obtain the parameters describing the initial displacements from the data of two available buoys located near the Japanese coast. Some of these parameters are reasonably well-known already, these include location of the hypocenter, length, width, and to some extent depth. Other parameters such as uplift are more difficult to estimate. In these tests we estimate the location of the initial displacement. The prior cuts off all parameters which would lead to an initial displacement which is too close to the domain boundary. Figure 6.8 shows the cut-off values used.

In order to compute a likelihood of a given set of parameters given the simulation results we use a weighted average of the maximal wave height and the time at which it is reached. The likelihood is given by a normal distribution $\mathcal{N}(\mu, \Sigma)$ with mean μ given by maximum wave height $\max\{h\}$ and the time t at which it is reached for the the two DART buoys 21418 and 21419 [2]. The covariance matrix Σ depends on

[2]This data can be obtained from NDBC `https://www.ndbc.noaa.gov/`

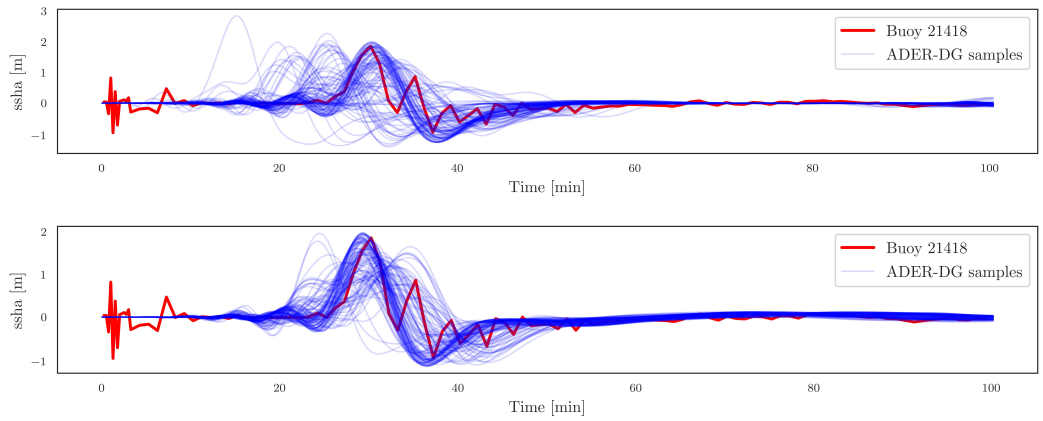


Fig. 6.9: Plot of Sea surface height anomaly [ssha] for samples taken from level 0 (top) and 1 (bottom) compared to NDBC data at buoy 21418.

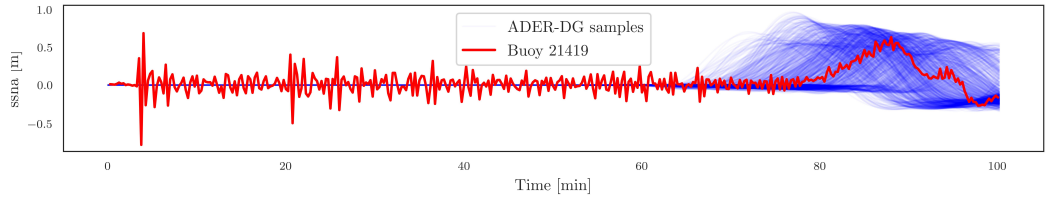


Fig. 6.10: Plot of Sea surface height anomaly [ssha] for samples taken from level 0 (top) and 1 (bottom) compared to NDBC data at buoy 21419.

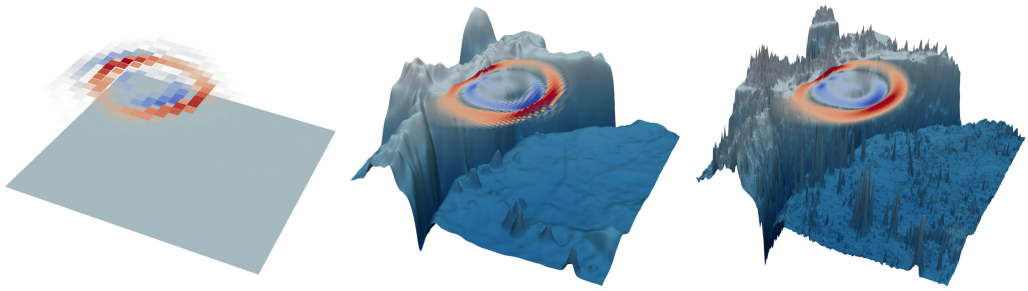


Fig. 6.11: Illustration of bathymetry smoothing and mesh coarsening for tsunami model.

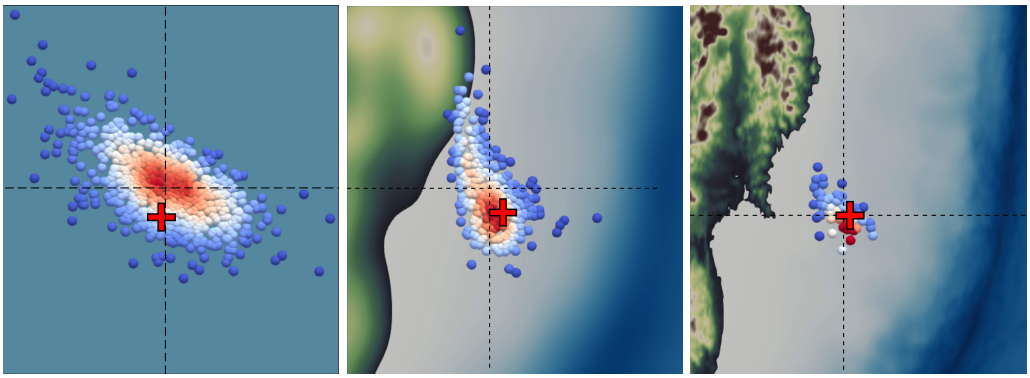


Fig. 6.12: The three-level tsunami test case, each point represents an accepted sample at level $l = 0, 1, 2$ (from left to right). The dashed lines show the expected value $\mathbb{E}(Q_0)$ or $\mathbb{E}(Q_0) + \sum_l \mathbb{E}[Q_l - Q_{l-1}]$ with the true origin $(0, 0)$ in red for reference.

the level, but not the probe point. Figures 6.9 and 6.10 shows the data and samples of level 0 and 1. Table 6.7 gives the values of μ and the diagonal entries of Σ for all three levels. Alternative likelihood functions, such as a quadratic average of multiple buoys could also be used, see e.g. [Beh+10].

We set up a sequence of three models with increasing accuracy shown as in Figure 6.11. In the first model bathymetry is approximated only by a depth average over the entire domain. Since no calculations of wetting and drying are needed this model is computed purely with a DG method of order 2. The second and third models further include a finite volume subcell limiter allowing for wetting and drying. The second model uses smoothed bathymetry data and the third uses the full bathymetry data. The main advantage of using smoothed data is that the FV subcell limiter is needed in fewer cells. The models use 25, 79 and 241 order two elements in each direction respectively. This model hierarchy demonstrates that not only mesh refinement and coarsening, but also model specific optimizations can be used to exploit multilevel MCMC.

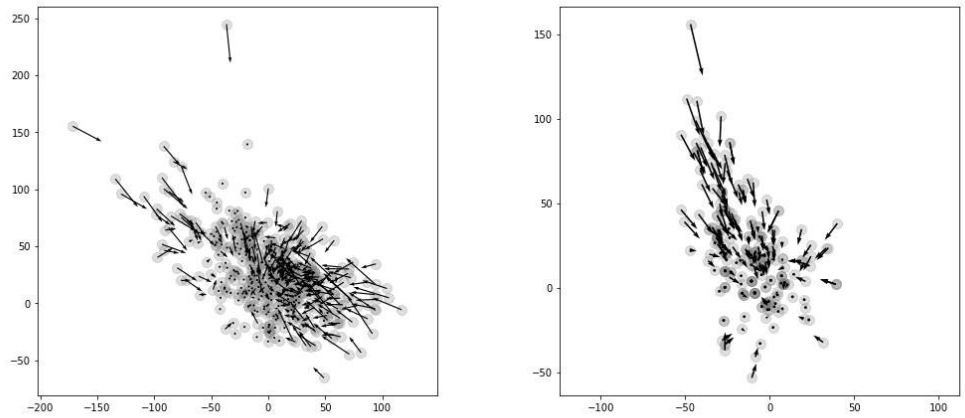


Fig. 6.13: Visualization of samples used to estimate the corrections between levels in eq. (3.1). Left: Correction between levels zero and one. Right: Correction between levels one and two. Dots and origins of arrows indicate coarse samples. The arrows point towards their corresponding fine samples. Note that the arrow's length is scaled down to a factor of .15 for clarity. Coarse proposals that were accepted by the fine chain, leading to identical coarse and fine samples here, appear as simple dots instead of arrows.

6.3.2 Results

As a second example of the MLMCMC method we invert data from the Tohoku tsunami. We find the parameter distribution describing the initial displacements from the data of two available buoys located near the Japanese coast.The MLMCMC method with three levels computes 800 samples on level 0, 450 on level 1 and 240 on level 2 with a subsampling rate of 25 on level 0 and 5 on level 1.

These tests were run on up to 72 Intel Skylake Xeon Platinum 8174 nodes of SuperMUC-ng consisting of 48 cores each. The tests were run using Intel's TBB for parallelisation over the 48 cores of each node and MPI for parallelisation across nodes. Each ExaHyPE run used exactly one full node. The runtime for each forward model evaluation was on average 7.38 seconds on level 0, 97.3 seconds on level 1 and 438.1 seconds on level 2. These runtimes have a large variablility as the model's timestep depends on the uncertain parameters, making it a challenging test for the scheduling infrastructure. In total $61,250$ level 0, 34500 level 1 and 240 level 2 forward model evaluations were required for this test.

Again, due to constant parameter dimension across levels, we only need to choose a proposal density for the coarsest level. Like before, we choose Adaptive Metropolis

lvl l	t_l[s]	ρ_l	$\mathbb{V}[Q_0]$ or $\mathbb{V}[Q_l - Q_{l-1}]$		$\mathbb{E}[Q_0]+$ $\sum_{k=1}^{l} \mathbb{E}[Q_k - Q_{k-1}]$	
0	7.38	25	1984.09	1337.42	3.61	27.96
1	97.3	5	1592.17	1523.18	−12.29	23.39
2	438.1	0	340.56	938.53	−5.46	0.12

Tab. 6.8: Multilevel properties of the tsunami model. For each level l of mesh, we show the computational cost t_l and chosen subsampling rate ρ_l. We show variance and expected values for both components of Q.

[HS98; HST01] provided by MIT Uncertainty Quantification Library (MUQ). As initial prior we set $\mathcal{N}(0, 10I)$ and update every 100 steps.

Figure 6.12 shows the resulting samples on level 0 and 1. The expected values $\mathbb{E}(Q_0)$, and $\mathbb{E}(Q_l - Q_{l-1})$ are shown as dashed line. The red marker shows the point $(0, 0)$. The point $(0, 0)$ is the position of the initial displacements as estimated in [Gal+14].

In order to illustrate how MLMCMC links coarser and finer level posteriors in order to obtain fine level corrections, fig. 6.13 shows how samples on level l relate to the level $l - 1$ samples that served as their respective coarse proposals. The result can be thought of as a transformation between corresponding coarser and finer distributions, even though only in a non-deterministic sense. Since in this application we choose the QOI to be the uncertain parameter itself, the estimate of the terms in the telescoping sum (eq. (3.1)) actually correspond to the mean of the corrections displayed here.

Variances and expected values are given in Table 6.8. The relatively cheap samples on level 0 provide a good initial estimate of the posterior, which are improved by the more expensive models utilizing the full bathymetry data. In contrast to the Poisson case, we do not observe variance reduction across levels. This is somewhat expected, as the modified bathymetry does not permit the construction of a level hierarchy fulfilling the theoretical assumptions made by MLMCMC based on a priori error estimates. We do, however, still have the benefit of well-informed proposals on finer levels driven by coarser level chains.

From this application, we conclude that the parallelization strategy proposed for MLMCMC and the corresponding implementation in MUQ can successfully be employed in solving large-scale real-world inverse problems.

7 Conclusion

As part of this dissertation, a parallelization strategy for Multilevel Markov Chain Monte Carlo (MLMCMC) and Multiindex Markov Chain Monte Carlo (MIMCMC) was developed in chapter 3, and implemented as part of the MIT Uncertainty Quantification Library (MUQ) software project in chapter 5. The implementation is both modular and highly scalable, overcoming inherent data dependencies. Numerous software components can be reused within the framework, and arbitrary forward models may be employed. The resulting methods allow solving a wide range of Uncertainty Quantification (UQ) problems, and are specifically designed to handle computationally challenging forward models. In addition, a new strategy for handling proposals in MIMCMC in analogy to the MLMCMC method from [Dod+19b] is proposed in section 3.4.

Coarsening strategies providing suitable model hierarchies for MLMCMC and MIMCMC are discussed in chapter 4. For multiscale Partial Differential Equation (PDE) models, section 4.2 presents a new theoretical link between robust preconditioners based on eigenproblems and Localized Model Order Reduction (LMOR). Consequently, methods from both fields can be exchanged. Using theoretical results from LMOR, it was proven that the Generalized Eigenproblems in the Overlaps (GenEO) space may serve as a model order reduction space. On the other hand, a randomized eigensolver used originally in LMOR could successfully be applied to the preconditioner setting (section 4.2.4). Further, an online/offline approach is proposed in section 4.4, allowing the reuse of large parts of the LMOR basis across runs. Together with an extension of the GenEO implementation in the Distributed and Unified Numerics Environment (DUNE) to overlaps generated algebraically (section 4.3), this paves the way for future UQ applications at a fraction of the computational cost.

Excellent parallel scalability and model-agnosticity of the algorithms proposed are demonstrated in the applications in chapter 6. In particular, the tsunami application in section 6.3 shows that the parallelized MLMCMC method allows for solving realistic inverse UQ problems on a scale where forward models themselves already pose significant computational challenges and require High Performance Computing (HPC) environments.

7.1 Future Work

MIMCMC and online/offline coarse approximation in UQ applications

Future work further investigating the advanced MIMCMC method introduced in section 3.4 is planned. In particular, applications to real-world models were unfortunately out of scope due to time constraints.

Likewise, work towards coarse approximation reusing local basis functions across multiple runs in an online/offline fashion (section 4.4) has progressed to a point where method and implementation are ready. Exploiting the online phase for low-cost coarse samples in an MLMCMC method and solving a UQ problem on multiscale a PDE is planned for a subsequent step. This will be carried out as part of the CerTest project [@Tho+].

Dynamic load balancing for heterogeneously parallelized models

The load balancing strategy for parallelized MLMCMC and MIMCMC presented in section 5.5 is currently limited to models parallelized across the same number of ranks for each level. Nevertheless, it has proven to be effective in the applications in chapter 6, effectively compensating for non-optimal initial load balancing as supplied by the user and ultimately yielding excellent scalability in large-scale tests (see section 6.1).

In a next step, in order to support heterogeneous model parallelization across levels, the load estimate per level could be adjusted to take into account the number of processes per model. Then, assuming integer factors between the numbers of ranks used in parallel models, assigning groups of processes to other levels should easily be achievable within the existing architecture.

For lack of a suitable application, this step was not undertaken yet, as it likely requires some more in-depth testing to ensure reliability. When applying the online/offline approach in section 4.4, this will likely play a crucial role, as in that case large-scale fine models will have to be balanced with sequential coarse models.

Randomized eigensolvers for coarse approximation in Markov Chain Monte Carlo (MCMC)

The randomized eigensolver used in section 4.2.4 turned out to be a viable option in practical applications of the GenEO preconditioner. Besides a slight efficiency gain in the scenarios tested, the main improvement over classic iterative eigensolvers turned out to be a gain in robustness.

A far more promising setting regarding run time gains however is an application in an MCMC setting: When multiple subsequent solves of nearly identical PDEs are executed, the GenEO basis from previous runs may be reused and only augmented for the new problem at hand. The randomized approach has the significant advantage over iterative solvers that additional basis functions can easily be added to an existing basis approximation. Therefore subsequent minimal basis updates are easily possible, avoiding full recomputations of the computationally expensive eigenproblem.

Model abstraction in software

Many UQ algorithms, in particular MCMC type methods as discussed in this thesis, simply assume a forward model in terms of a function $F : \mathbb{R}^n \to \mathbb{R}^m$ mapping parameters onto model predictions. The design choices in the implementations presented in chapter 5 exploit this, making use of the abstract model interface in the MUQ. As a result, the algorithms implemented can be used with arbitrary (and possibly already existing) user model codes.

While developing the applications in chapter 6 however, it turned out that coupling complex model codes to advanced UQ codes may pose significant difficulties even despite having a general-purpose interface. Numerous complications can arise already in compiling such a software stack. Further, model codes may not be intended to be called by other software in the first place.

For example, in the tsunami application in section 6.3 we made use of the Exascale Hyperbolic PDE Engine (ExaHyPE) package for the forward model. It is intended to be run for individual simulation runs, and in that use case is in fact easy to use and highly efficient. In its current form, however, it is not intended to be used as a software library. Overcoming resulting fundamental architectural limitations took an enormous technical effort, binding resources that we would have liked to invest in the design of the actual UQ problem to be solved.

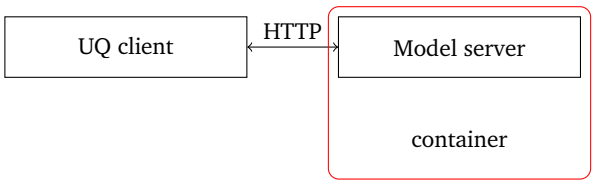


Fig. 7.1: Network and container based coupling of UQ and model codes.

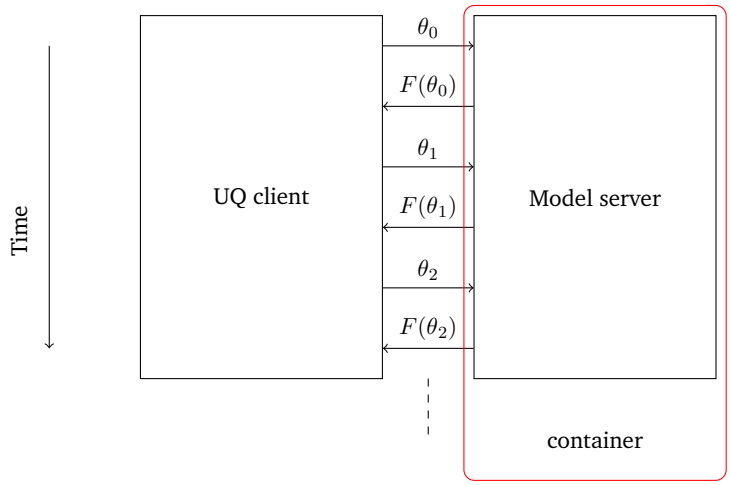


Fig. 7.2: Sequence of messages passed between client and server in case of an MCMC algorithm generating samples θ_i.

It is therefore a very promising idea to truly decouple model and UQ software stacks in order to leverage the model abstraction promised by UQ algorithm and theory.

As an alternative to a "hard" coupling in the sense of compiling model and UQ codes together, an approach based on containerization and network communication was developed. While initially intended as a "backup" solution, it turns out to have a wider practical value.

The key point is to have the model code ("server") and UQ code ("client") exchange information via a simple HTTP-based network protocol. Since the amount of data to be exchanged consists of a typically small parameter vector and a resulting model prediction vector for each (typically expensive) model evaluation, the overhead compared to exchanging that information directly through shared memory is insignificant for all but the most exotic applications.

At this point, server and client can be compiled entirely independently; in fact, since only support for the simple HTTP protocol is required on both ends, they may even use completely unrelated programming languages or dependencies.

In a next step, such a model may be run and distributed inside a container, for example a docker [Mer14] or singularity [KSB17] container. The resulting software setup is illustrated in fig. 7.1, and an exemplary timeline of messages passed in fig. 7.2. This makes it possible to

- easily deploy complex models on HPC systems with container support as well as public cloud infrastructure while working in a standard and locally testable software environment,
- provide an entire library of pre-built models to the UQ community for testing and benchmarking purposes,
- and easily exchange complex models between collaborating researchers, completely bypassing software setup. Consequently, development can be arranged by exploiting separation of concerns: Model experts only have to provide the interface, while UQ developers need no understanding of the details of the underlying model.

This ongoing work should further contribute to bridging the gap between advanced models and advanced UQ methods.

Bibliography

[Aln+15] Martin S. Alnæs, Jan Blechta, Johan Hake, et al. “The FEniCS Project Version 1.5”. In: *Archive of Numerical Software* 3.100 (2015) (cit. on p. 67).

[Alz+18] G. Alzetta, D. Arndt, W. Bangerth, et al. “The `deal.II` Library, Version 9.0”. In: *Journal of Numerical Mathematics* 26.4 (2018), pp. 173–183 (cit. on p. 67).

[BL11] I Babuška and R Lipton. *Optimal local approximation spaces for generalized finite element methods with application to multiscale problems*. English. Philadelphia: SIAM Publications, 2011, pp. 373–406 (cit. on p. 79).

[BHM10] P Bastian, F Heimann, and S Marnach. “Generic implementation of finite element methods in the Distributed and Unified Numerics Environment (Dune).” In: *Kybernetika* 46.2 (2010), pp. 294–315 (cit. on pp. 67, 116).

[Bas12] Peter Bastian. *Lecture notes: Numerical Methods for Partial Differential Equations*. Sept. 2012 (cit. on pp. 8, 13, 18).

[Bau01] Heinz Bauer. *Measure and integration theory*. eng. De Gruyter studies in mathematics ARRAY(0x5573a0f975d0). Includes bibliographical references and indexes. Berlin ; New York: de Gruyter, 2001, XVI, 230 S. (Cit. on p. 21).

[Beh00] Ehrhard Behrends. *Introduction to Markov chains. with special emphasis on rapid mixing*. eng. Advanced lectures in mathematics. Braunschweig ; Wiesbaden: Vieweg, 2000, IX, 232 S. (Cit. on p. 23).

[Beh+10] J. Behrens, A. Androsov, A. Y. Babeyko, et al. “A new multi-sensor approach to simulation assisted tsunami early warning”. In: *Natural Hazards and Earth System Sciences* 10.6 (2010), pp. 1085–1100 (cit. on pp. 126, 130).

[Bes+08] Alexandros Beskos, Gareth Roberts, Andrew Stuart, and Jochen Voss. “MCMC METHODS FOR DIFFUSION BRIDGES”. In: *Stochastics and Dynamics* 08.03 (2008), pp. 319–350. eprint: `https://doi.org/10.1142/S0219493708002378` (cit. on p. 49).

[BS07] Dietrich Braess and Larry L. Schumaker. *Finite elements. theory, fast solvers, and applications in elasticity theory*. eng. 3. ed. Cambridge [u.a.]: Cambridge University Press, 2007, XVII, 365 S (cit. on pp. 8, 13, 16, 18).

[Bro+11] Steve Brooks, Andrew Gelman, Galin L. Jones, and Xiao-Li Meng, eds. *Handbook of Markov chain Monte Carlo*. eng. A Chapman & Hall book. Literaturangaben. Boca Raton ; London [u.a.]: CRC Press, Taylor & Francis, 2011, XXV, 592 S. (Cit. on p. 23).

[Buh19] Andreas Buhr. “Towards Automatic and Reliable Localized Model Order Reduction”. PhD thesis. Westfälische Wilhelms-Universität Münster, 2019 (cit. on pp. 69, 76, 79, 84).

[Buh+19] Andreas Buhr, Laura Iapichino, Mario Ohlberger, et al. “Handbook on Model Order Reduction”. In: submitted for publication. Walter De Gruyter GmbH, 2019+. Chap. Localized model reduction for parameterized problems (cit. on p. 76).

[BS18] Andreas Buhr and Kathrin Smetana. “Randomized Local Model Order Reduction”. In: *SIAM Journal on Scientific Computing* 40.4 (2018), A2120–A2151 (cit. on p. 84).

[But+20a] R. Butler, T. Dodwell, A. Reinarz, et al. “High-performance dune modules for solving large-scale, strongly anisotropic elliptic problems with applications to aerospace composites”. In: *Computer Physics Communications* 249 (2020), p. 106997 (cit. on p. 75).

[But+20b] R. Butler, T. Dodwell, A. Reinarz, et al. “High-performance dune modules for solving large-scale, strongly anisotropic elliptic problems with applications to aerospace composites”. In: *Computer Physics Communications* 249 (2020), p. 106997 (cit. on pp. 85, 95).

[CS99] XC Cai and M Sarkis. “A restricted additive Schwarz preconditioner for general sparse linear systems”. English. In: *SIAM JOURNAL ON SCIENTIFIC COMPUTING* 21.2 (1999), pp. 792–797 (cit. on p. 88).

[Cha+03] T Chartier, R. D. Falgout, V. E. Henson, et al. *Spectral AMGe (ρAMGe)*. English. Philadelphia: Society for Industrial and Applied Mathematics, 2003, p. 1 (cit. on p. 67).

[CF05] J. Christen and Colin Fox. “Markov chain Monte Carlo Using an Approximation”. In: *Journal of Computational and Graphical Statistics - J COMPUT GRAPH STAT* 14 (Dec. 2005), pp. 795–810 (cit. on p. 53).

[CR72] P.G. Ciarlet and P.-A. Raviart. “Interpolation theory over curved elements, with applications to finite element methods”. In: *Computer Methods in Applied Mechanics and Engineering* 1.2 (1972), pp. 217–249 (cit. on p. 16).

[CK19] David Colton and Rainer Kress. “Inverse Acoustic and Electromagnetic Scattering Theory”. In: 4th ed. New York: Springer-Verlag, 2019 (cit. on p. 118).

[Cot+13] S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. “MCMC Methods for Functions: Modifying Old Algorithms to Make Them Faster”. In: *Statist. Sci.* 28.3 (Aug. 2013), pp. 424–446 (cit. on p. 49).

[CDS19] Tiangang Cui, Gianluca Detommaso, and Robert Scheichl. “Multilevel Dimension-Independent Likelihood-Informed MCMC for Large-Scale Inverse Problems”. In: (Oct. 2019) (cit. on p. 49).

[CLM16] Tiangang Cui, Kody J.H. Law, and Youssef M. Marzouk. “Dimension-independent likelihood-informed MCMC”. In: *Journal of Computational Physics* 304 (2016), pp. 109–137 (cit. on p. 49).

[Dav+21] Andrew D. Davis, Youssef Marzouk, Aaron Smith, and Natesh Pillai. *Rate-optimal refinement strategies for local approximation MCMC*. 2021. arXiv: 2006.00032 [stat.CO] (cit. on p. 98).

[DN97] C. Dietrich and G. Newsam. "Fast and Exact Simulation of Stationary Gaussian Processes through Circulant Embedding of the Covariance Matrix". In: *SIAM J. Sci. Comput.* 18 (1997), pp. 1088–1107 (cit. on p. 116).

[Dod08] Yadolah Dodge. *The concise encyclopedia of statistics. with 247 tables*. eng. Springer reference. Literaturverz. S. [597] - 616. [New York, NY]: Springer, 2008, VIII, 616 S. (Cit. on p. 27).

[Dod+19a] T. J Dodwell, C Ketelsen, R Scheichl, and A. L Teckentrup. "ERRATUM: A Hierarchical Multilevel Markov Chain Monte Carlo Algorithm with Applications to Uncertainty Quantification in Subsurface Flow". English. In: *SIAM/ASA journal on uncertainty quantification* 7.4 (2019), pp. 1398–1399 (cit. on p. 55).

[Dod+15] Tim Dodwell, Chris Ketelsen, Robert Scheichl, and Aretha Teckentrup. "A Hierarchical Multilevel Markov Chain Monte Carlo Algorithm with Applications to Uncertainty Quantification in Subsurface Flow". In: (Aug. 2015) (cit. on pp. 49, 50, 53, 120).

[Dod+19b] Tim Dodwell, Chris Ketelsen, Robert Scheichl, and Aretha Teckentrup. "Multilevel Markov Chain Monte Carlo". In: *SIAM Review* 61 (Jan. 2019), pp. 509–545 (cit. on pp. 41, 49, 50, 53, 55, 58–61, 120, 133).

[Dua+87] Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. "Hybrid Monte Carlo". In: *Physics Letters B* 195.2 (1987), pp. 216–222 (cit. on p. 49).

[Dum+08] Michael Dumbser, Dinshaw S. Balsara, Eleuterio F. Toro, and Claus-Dieter Munz. "A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes". In: *Journal of Computational Physics* 227.18 (2008), pp. 8209–8253 (cit. on p. 127).

[DL16] Michael Dumbser and Raphaël Loubère. "A simple robust and accurate a posteriori sub-cell finite volume limiter for the discontinuous Galerkin method on unstructured meshes". In: *Journal of Computational Physics* 319 (2016), pp. 163–199 (cit. on p. 127).

[Gal+14] P. Galvez, J.-P. Ampuero, L. A. Dalguer, S. N. Somala, and T. Nissen-Meyer. "Dynamic earthquake rupture modelled with an unstructured 3-D spectral element method applied to the 2011 M9 Tohoku earthquake". In: *Geophysical Journal International* 198.2 (June 2014), pp. 1222–1240. eprint: https://academic.oup.com/gji/article-pdf/198/2/1222/1651569/ggu203.pdf (cit. on pp. 127, 132).

[GS91] Roger G. Ghanem and Pol D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Berlin, Heidelberg: Springer-Verlag, 1991 (cit. on p. 48).

[Gil08] Michael B. Giles. "Multilevel Monte Carlo Path Simulation". In: *Oper. Res.* 56.3 (May 2008), pp. 607–617 (cit. on pp. 49–51).

[HS98] Heikki Haario and Eero Saksman. “Adaptive Proposal Distribution for Random Walk Metropolis Algorithm”. In: *Computational Statistics* 14 (July 1998) (cit. on pp. 49, 124, 132).

[HST01] Heikki Haario, Eero Saksman, and Johanna Tamminen. “An Adaptive Metropolis Algorithm”. In: *Bernoulli* 7 (Apr. 2001) (cit. on pp. 49, 124, 132).

[Haj+16] Abdul-Lateef Haji-Ali, Abdul-Lateef Haji-Ali, Fabio Nobile, et al. “Multi-index Monte Carlo: when sparsity meets sampling”. English. In: *Numerische Mathematik* 132.4 (2016), pp. 767–806 (cit. on pp. 61, 63, 68, 126).

[Has70] W. K Hastings. “Monte Carlo sampling methods using Markov chains and their applications”. English. In: *Biometrika* 57.1 (1970), pp. 97–109 (cit. on p. 35).

[Hei01] Stefan Heinrich. “Multilevel Monte Carlo Methods”. In: *Large-Scale Scientific Computing*. Ed. by Svetozar Margenov, Jerzy Waśniewski, and Plamen Yalamov. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 58–67 (cit. on pp. 49, 50).

[Hjo+10] Nils Lid Hjort, Chris Holmes, Peter Müller, and Stephen G. Walker, eds. *Bayesian nonparametrics*. eng. 1. publ. Cambridge series in statistical and probabilistic mathematics ARRAY(0x55ea7e392be0). Cambridge [u.a.]: Cambridge University Press, 2010, VIII, 299 S. (Cit. on p. 25).

[Hol07] Mark Holmes. *Introduction to Numerical Methods in Differential Equations*. Vol. 52. Jan. 2007 (cit. on p. 31).

[17] *Hypre - high performance preconditioners User’s Manual*. Tech. rep. Software Version: 2.11.2. `https://computation.llnl.gov/sites/default/files/public/hypre-2.11.2_usr_manual.pdf`: Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Mar. 2017 (cit. on p. 67).

[Jas+17] Ajay Jasra, Kengo Kamatani, Kody Law, and Yan Zhou. “A Multi-Index Markov Chain Monte Carlo Method”. In: *International Journal for Uncertainty Quantification* 8 (Mar. 2017) (cit. on p. 61).

[Kan07] Guido Kanschat. *Discontinuous Galerkin methods for viscous incompressible flow*. eng. 1. ed. Advances in numerical mathematics. Wiesbaden: Teubner Research, 2007, 183 S. (Cit. on p. 16).

[Kit95] Peter K. Kitanidis. “Quasi-Linear Geostatistical Theory for Inversing”. In: *Water Resources Research* 31.10 (1995), pp. 2411–2419. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/95WR01945` (cit. on p. 49).

[Kle+17] Ole Klein, Olaf A. Cirpka, Peter Bastian, and Olaf Ippisch. “Efficient geostatistical inversion of transient groundwater flow using preconditioned nonlinear conjugate gradients”. In: *Advances in Water Resources* 102 (2017), pp. 161–177 (cit. on p. 49).

[KSB17] Gregory M Kurtzer, Vanessa Sochat, and Michael W Bauer. “Singularity: Scientific containers for mobility of compute”. English. In: *PloS one* 12.5 (2017). Ed. by Attila Gursoy, e0177459–e0177459 (cit. on p. 137).

[LK14] Jonghyun Lee and P. Kitanidis. “Large-scale hydraulic tomography and joint inversion of head and tracer data using the Principal Component Geostatistical Approach (PCGA)”. In: *Water Resources Research* 50 (July 2014) (cit. on p. 49).

[LSY97] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users Guide: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods.* 1997 (cit. on p. 85).

[LeV02] Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002 (cit. on p. 11).

[LGB11] Randall J. LeVeque, David L. George, and Marsha J. Berger. “Tsunami modelling with adaptively refined finite volume methods”. In: *Acta Numerica* 20 (2011), pp. 211–289 (cit. on pp. 126, 127).

[Lyk+20] Mikkel B. Lykkegaard, Grigorios Mingas, Robert Scheichl, Colin Fox, and Tim J. Dodwell. *Multilevel Delayed Acceptance MCMC with an Adaptive Error Model in PyMC3*. 2020. arXiv: `2012.05668 [stat.CO]` (cit. on p. 59).

[Mad+21] Elizabeth Madden, Michael Bader, Jörn Behrens, et al. “Linked 3D modeling of megathrust earthquake-tsunami events: from subduction to tsunami run up”. en. In: *Geophysical Journal International* 224.1 (Oct. 2021), pp. 487–516 (cit. on p. 127).

[Mar+16] Youssef Marzouk, Tarek Moselhy, Matthew Parno, and Alessio Spantini. “Sampling via Measure Transport: An Introduction”. In: Jan. 2016, pp. 1–41 (cit. on p. 98).

[MX09] Youssef Marzouk and Dongbin Xiu. “A Stochastic Collocation Approach to Bayesian Inference in Inverse Problems”. In: *PRISM: NNSA Center for Prediction of Reliability, Integrity and Survivability of Microsystems* 6 (Oct. 2009) (cit. on p. 49).

[McC18] Ryan G. McClarren. *Uncertainty Quantification and Predictive Computational Science. A Foundation for Physical Scientists and Engineers*. eng. SpringerLink : Bücher. Cham: Springer, 2018, Online–Ressource (XVII, 345 Seiten) (cit. on p. 29).

[Mer14] Dirk Merkel. “Docker: lightweight linux containers for consistent development and deployment”. In: *Linux journal* 2014.239 (2014), p. 2 (cit. on p. 137).

[Met+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. “Equation of State Calculations by Fast Computing Machines”. In: *The Journal of Chemical Physics* 21.6 (1953), pp. 1087–1092 (cit. on p. 35).

[PWG16] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. “Optimal Model Management for Multifidelity Monte Carlo Estimation”. English. In: *SIAM journal on scientific computing* 38.5 (2016), A3163–A3194 (cit. on p. 69).

[RDB18] Leonhard Rannabauer, Michael Dumbser, and Michael Bader. “ADER-DG with a-posteriori finite-volume limiting to simulate tsunamis in a parallel adaptive mesh refinement framework”. In: *Computers and Fluids* 173 (2018), pp. 299–306 (cit. on p. 127).

[Ran+18] Leonhard Rannabauer, Stefan Haas, Dominic Etienne Charrier, Tobias Weinzierl, and Michael Bader. “Simulation of tsunamis with the exascale hyperbolic PDE engine ExaHyPE”. en. In: *Environmental Informatics: Techniques and Trends. Adjunct Proceedings of the 32nd edition of the EnviroInfo.* Shaker Verlag, Sept. 2018 (cit. on p. 127).

[Rei+18] Anne Reinarz, Tim Dodwell, Tim Fletcher, et al. “Dune-composites - A new framework for high-performance finite element modelling of laminates”. In: *Composite Structures* 184 (2018), pp. 269–278 (cit. on pp. 85, 88, 90, 95).

[RC10] Christian P. Robert and George Casella. *Monte Carlo statistical methods*. eng. 2. ed., softcover reprint of the hardcover 2. ed. 2004. Springer texts in statistics. Literaturverz. S. [591] - 622. New York, NY: Springer New York, 2010, XXX, 645 S. (Cit. on pp. 41, 42, 44).

[Rud12] Daniel Rudolf. “Explicit error bounds for Markov chain Monte Carlo”. In: *Dissertationes Mathematicae* 485 (2012), pp. 1–93 (cit. on p. 44).

[RS15] Daniel Rudolf and Björn Sprungk. “On a Generalization of the Preconditioned Crank–Nicolson Metropolis Algorithm”. In: *Foundations of Computational Mathematics* (Apr. 2015) (cit. on p. 49).

[Saa07] Youcef Saad. *Iterative methods for sparse linear systems*. eng. 2. ed., [Nachdr.] Philadelphia, PA: SIAM, Society for Industrial and Applied Mathematics, 2007, XVIII, 528 S. (Cit. on p. 14).

[SF09] Tatsuhiko Saito and Takashi Furumura. “Three-dimensional tsunami generation simulation due to sea-bottom deformation and its interpretation based on the linear theory”. In: *Geophysical Journal International* 178.2 (Aug. 2009), pp. 877–888. eprint: `https://academic.oup.com/gji/article-pdf/178/2/877/5917374/178-2-877.pdf` (cit. on p. 127).

[SZ21] Robert Scheichl and Jakob Zech. *Lecture notes: Numerical Methods for Bayesian Inverse Problems*. Mar. 2021 (cit. on p. 19).

[See+21] Linus Seelinger, Anne Reinarz, Leonhard Rannabauer, et al. “High Performance Uncertainty Quantification with Parallelized Multilevel Markov Chain Monte Carlo”. In: *Supercomputing 21* to appear (2021) (cit. on pp. 47, 97, 115).

[SRS20] Linus Seelinger, Anne Reinarz, and Robert Scheichl. “A High-Performance Implementation of a Robust Preconditioner for Heterogeneous Problems”. In: *Parallel Processing and Applied Mathematics*. Ed. by Roman Wyrzykowski, Ewa Deelman, Jack Dongarra, and Konrad Karczewski. Cham: Springer International Publishing, 2020, pp. 117–128 (cit. on pp. 3, 70, 75, 85, 93).

[SP16a] Kathrin Smetana and Anthony T Patera. “Optimal Local Approximation Spaces for Component-Based Static Condensation Procedures”. English. In: *SIAM journal on scientific computing* 38.5 (2016), A3318–A3356 (cit. on p. 79).

[SP16b] Kathrin Smetana and Anthony T. Patera. “Optimal Local Approximation Spaces for Component-Based Static Condensation Procedures”. In: *SIAM Journal on Scientific Computing* 38.5 (2016), A3318–A3356 (cit. on p. 79).

[Smi86] Vladimir I. Smirnov. ger. 16. Aufl. Hochschulbücher für Mathematik ARRAY(0x55f2476862c8). Berlin: Dt. Verl. d. Wiss., 1986, 618 S. (Cit. on p. 9).

[SBG96] Barry F. Smith, Petter E. Bjørstad, and William Gropp. *Domain decomposition. parallel multilevel methods for elliptic partial differential equations*. eng. Includes bibliographical references. Cambridge [u.a.]: Cambridge Univ. Press, 1996, XII, 224 S. (Cit. on pp. 67, 70).

[Spi+14] N Spillane, V Dolean, P Hauret, et al. *Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps*. English. Berlin/Heidelberg: Springer Berlin Heidelberg, 2014, pp. 741–770 (cit. on pp. 70, 73–76).

[Spi+11] Nicole Spillane, Victorita Dolean, Patrice Hauret, et al. *A robust two-level domain decomposition preconditioner for systems of PDEs*. English. Elsevier B.V, 2011, pp. 1255–1259 (cit. on p. 74).

[Sul15] T. J. Sullivan. *Introduction to Uncertainty Quantification*. eng. Springer eBook Collection. Cham: Springer, 2015, Online–Ressource (XII, 342 p. 28 illus, online resource) (cit. on p. 19).

[TW05] Andrea Toselli and Olof Widlund. *Domain decomposition methods - algorithms and theory*. eng. Springer series in computational mathematics. Berlin ; Heidelberg [u.a.]: Springer, 2005, XV, 450 S. (Cit. on pp. 70, 72).

[Wol04] Ulli Wolff. “Monte Carlo errors with less errors”. In: *Computer Physics Communications* 156.2 (2004), pp. 143–153 (cit. on p. 24).

[Xiu15] Dongbin Xiu. “Stochastic Collocation Methods: A Survey”. In: Jan. 2015, pp. 1–18 (cit. on p. 49).

[XK03] Dongbin Xiu and George Em Karniadakis. “Modeling Uncertainty in Flow Simulations via Generalized Polynomial Chaos”. In: *J. Comput. Phys.* 187.1 (May 2003), pp. 137–167 (cit. on p. 48).

[YH02] Ulrike Meier Yang and Van Emden Henson. “BoomerAMG: A parallel algebraic multigrid solver and preconditioner”. In: *Applied Numerical Mathematics* 41.1 (2002), pp. 155–177 (cit. on p. 67).

Webpages

[@Par+] Matthew Parno, Andrew Davis, Linus Seelinger, and Youssef Marzouk. *MUQ project website*. URL: http://muq.mit.edu/ (cit. on pp. 97, 106).

[@Tho+] Ole Thomsen et al. *CerTest project website*. URL: https://www.composites-certest.com/ (cit. on p. 134).

Acronyms

ADER-DG Arbitrary-high-order-DERivative Discontinuous Galerkin Method. 127

CG Conjugate Gradient. 94

DOF Degree of freedom. 90–92, 154

DUNE Distributed and Unified Numerics Environment. 3, 68, 90, 91, 116, 133

ESS Effective Sample Size. 24, 125, 126

ExaHyPE Exascale Hyperbolic PDE Engine. 97, 109, 135

FE Finite element. 8, 13, 16, 18, 43, 53, 70, 71, 91, 93, 123

GenEO Generalized Eigenproblems in the Overlaps. 2–4, 68–70, 80, 83, 84, 90–94, 133, 135, 154

HPC High Performance Computing. v, 2, 4, 5, 97–99, 115, 120, 133, 137

i.i.d. Independent and identically distributed. 31, 33, 43, 44, 54–56, 58

KL Karhunen Loève. 116, 120, 123

LMOR Localized Model Order Reduction. v, 4, 5, 68, 69, 80, 83, 84, 94, 133, 151, 152

MAP Maximum a posteriori probability. 48, 49

MC Monte Carlo. 3, 32, 34, 38, 44, 45, 47–49, 51, 52, 105, 150

MCMC Markov Chain Monte Carlo. 3, 23, 24, 34, 35, 39, 41–45, 47, 49, 52, 53, 56–58, 60, 61, 63, 69, 98–100, 105–108, 111, 116, 124, 130, 135, 136, 153, 156

MHMCMC Metropolis-Hastings Markov Chain Monte Carlo. 35, 36, 39, 41, 49, 55, 56

MIMC Multiindex Monte Carlo. 61, 63, 126

MIMCMC Multiindex Markov Chain Monte Carlo. 2, 4, 5, 47, 61, 62, 64, 65, 67, 97, 105, 107, 122, 124, 126, 133, 134, 153

MLMC Multilevel Monte Carlo. 2, 49–52, 60, 61, 97, 111

MLMCMC Multilevel Markov Chain Monte Carlo. 2, 4, 5, 47, 49, 50, 53, 56, 58, 60–62, 64, 65, 67, 68, 94, 97, 98, 105–107, 109–111, 115, 116, 120, 126, 131–134, 153, 155

MOR Model Order Reduction. 2, 3, 5, 6, 69

MPI Message Passing Interface. 108, 109

MSE Mean square error. 33, 34, 42, 45

MUQ MIT Uncertainty Quantification Library. v, 2, 3, 24, 97–99, 106, 109, 132, 133, 135

ODE Ordinary Differential Equation. 29–31, 33, 50, 51, 69

PDE Partial Differential Equation. 2–8, 11–13, 18, 29, 48, 50, 67–70, 80, 93, 94, 115, 123, 124, 133–135, 151, 152, 154, 155

QOI Quantity of Interest. 25, 26, 28, 30, 33, 37–39, 42, 43, 52, 118, 119, 123, 132, 153

RV Random Variable. 21–23, 25, 31

TBB Intel Threading Building Blocks. 109

UQ Uncertainty Quantification. v, 1–8, 18, 23, 25, 28–32, 35, 36, 38, 47, 48, 50, 53, 67–69, 94, 95, 97–99, 106, 115, 118, 133–137, 153, 156

Symbols: Uncertainty Quantification

$(\Omega, \mathcal{F}, \mathbb{P})$ Probability space. 21–23

F Forward model. 25–29, 31–35, 37, 38, 48, 50–52, 67, 135, 136, 150

K Transition kernel. 41

M Constant representing levels' accuracy, corresponds to discretization level. 59, 60

Pot Power set. 19

Q Quantity of interest. 26, 28, 30, 33, 39, 41–46, 52, 53, 58–63, 107, 118, 123–125, 150, 153

R Parameter dimension in multilevel methods. 59, 60

V Banach space. 21, 22

X Random variable. 21, 22

Y Difference between levels. 60

Y Correction term. 53, 55, 59

Ω Sample space. 19–23, 149

Θ Uncertain model parameter on coarser chain. 53–56, 59

α Multiindex. 61

$\hat{\theta}$ Estimate of uncertain model parameter. 27

$\mathbb{P}$ Probability measure. 20–23, 149

$\mathcal{F}$ Sample space. 19–23, 149

$\mathcal{L}$ Likelihood function. 27, 28, 35, 37

$\mathcal{P}(y)$ Measurement distribution. 27, 28, 35, 38

μ Measure. 19–22

ν Posterior distribution. 27, 28, 42–45, 52–55, 58–63, 124, 125

π_0 Prior density. 25–28, 31, 35, 37, 41, 58, 118

π Posterior density. 27, 35, 41, 53, 55, 56, 59

τ Integrated autocorrelation time. 24, 39, 44–46, 56, 58

θ Uncertain model parameter. 25–38, 40–43, 50, 51, 53–60, 63, 108, 117, 123, 124, 136, 155, 156

$\widehat{Y}^{MCMC}$ Estimator of correction term. 53–55, 60

$\widehat{F}$ Monte Carlo (MC) estimator. 32–34, 51

$\widehat{Q}$ Q estimator. 42–45, 53, 59, 60

$_*$ Pushforward symbol. 21, 26–28, 31

m Model parameter dimension. 25, 36, 41, 50

n Forward model output dimension. 25, 50

q Proposal distribution. 36, 38, 40, 41, 55, 56, 58, 59

y Measurement. 26–28, 35

Symbols: Partial Differential Equations and Localized Model Order Reduction

J^* Maximum number of domains ω_i^* at any point in Ω. 78, 79

N_{V_i} Number of local spaces in LMOR. 77–79, 82

P Orthogonal projection, P_X maps into X. 79, 80

R Range space of the transfer operator T. 79

S_i Intermediate space of localized training configuration. 78, 79, 151

S Source space of the transfer operator T. 79, 80

T^l Linear part of T. 79, 80

T_i^a Affine part of T_i. 78, 79

T_i^l Linear part of T_i. 78–80

T_i Affine linear compact operator, mapping from S_i to V_i. 78, 79, 82, 151

T Transfer operator. 151, 152

$V|_{\omega_i^*}$ Function space restricted to training domain. 78, 79

V_i Local function space in LMOR. 77–79, 151

V Function space. 77–79, 151

Ω PDE domain. 77–79, 151

α Coercivity constant of a. 13, 16, 79

$\bar{J}$ Number of orthogonal classes. 79

γ Continuity constant of a. 13, 79

$\mathcal{P}_{S_i}$ Mapping from full space V to local space S_i. 78, 79, 81, 82

$\mathcal{P}_{V_i}$ Mapping from full space V to local space V_i. 77–79, 81

ω_i^* Subdomain of localized training configuration. 78, 79, 82, 151

ω_i Subdomain of LMOR. 77–79

σ Singular value. 79, 80

$\widetilde{u}$ Reduced solution. 78, 79

$\widetilde{R}^n$ Reduced range space of the transfer operator T. 79

$\widetilde{V}_i$ Local function space. 79

$\widetilde{V}$ Reduced function space. 78, 79

a Bilinear form. 78

b Linear form. 78

d PDE dimension. 77

q Left singular vector. 79, 80

u Solution of variational problem. 78, 79, 82

v Right singular vector. 79, 80

List of Figures

Colophon

This thesis was typeset with LaTeX 2$_\varepsilon$. It uses a slightly modified version of the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant INST 35/1134-1 FUGG.

The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer SuperMUC-NG at Leibniz Supercomputing Centre (www.lrz.de).

Erratum: GenEO condition estimate in "Multiscale Methods for High Performance Uncertainty Quantification"

Linus Seelinger

December 21, 2021

The recapitulation of the condition estimate for additive two-level Schwarz methods in section 4.2.1 contains an error. While the restriction operators R_j and R_H used to define the application of the preconditioner are defined correctly, they are incorrectly used in the "stable splitting" assumption. This has implications on the formulation of some subsequent steps.

Stable splitting definition (p. 73)

The definition of stable splitting should instead read:

Definition 30 (Stable splitting). Assume a coarse space $V_H \subset V_h$ and per-subdomain subspaces $V_{h,0}(\Omega_j)$ as defined above. The decomposition of $v \in V_h$

$$v = \sum_{j=0}^{N} z_j, \qquad z_0 \in V_H, \qquad z_j \in V_{h,0}(\Omega_j) \text{ for } 1 \leq j \leq N,$$

is called C_0-stable iff

$$\|z_0\|_a^2 + \sum_{j=1}^{N} \|z_j\|_{a,\Omega_j}^2 \leqslant C_0^2 \|v\|_a^2.$$

Lemma 6 then correctly concludes a bound on the minimal eigenvalue of the preconditioned system if there exists a common C_0 such that all $v \in V_h$ permit a C_0-stable splitting.
(Note that the decomposition in Definition 30 is indeed somewhat analogous to the restriction operators R_j and R_H. However, $v = R_H v + \sum_j R_j v$ does not hold, at the very least due to subdomain overlaps. Further, Definition 30 only needs to ask for any suitable decomposition to exist, and requiring a particular one would be unnecessarily restrictive.)

Localized splitting (p. 74)

As a consequence, Lemma 7 also has to be adjusted slightly in order to match the correct decomposition:

Lemma 7 (Localized splitting).
Let there exist a constant C_1 fulfilling

$$\|z_j\|^2_{a,\Omega_j} \leqslant C_1 |v|^2_{a,\Omega_j}$$

for all $1 \leqslant j \leqslant N$. Then the decomposition $v = \sum_{j=0}^{N} z_j$ as above is C_0-stable with $C_0^2 = 2 + C_1 k_0 (2k_0 + 1)$.
Proof: Remains as before.

Modified eigenproblem (p. 76)

Definition 35 introduces a modified GenEO eigenproblem by replacing the right hand side bilinear form $a_{\Omega_j^o}$ (i.e. the one restricted to the overlap region) by the full a_{Ω_j}. A condition estimate nearly identical to the original GenEO result can then be shown for the modified eigenproblem by dropping the treatment of the interior $\Omega_j \setminus \Omega_j^o$ in the proof of (Lemma 3.21, Spillane 2014). The resulting localized splitting formulated with the correct decomposition now reads

$$\|z_j\|^2_{a,\Omega_j} \leq \frac{1}{\lambda^j_{m_j+1}} |v|^2_{a,\Omega_j}.$$

Apart from that change, the overall argument and resulting condition estimate remain. A more extensive analysis of several related GenEO-type eigenproblems can be found in [1]. Specifically, the one used here is defined in [1, equation 3.9a].

Localized splitting from LMOR (p. 83)

Lemma 11 shows how the approximation result from the localized model order reduction (LMOR) framework can in turn be applied to prove the stable splitting assumption in the abstract Schwarz theory. Now that the particular choice of decomposition is left open in the stable splitting assumption above, Lemma 11 needs to explicitly introduce it.

Lemma 11. We follow the stable splitting construction in GenEO analysis (Lemma 3.20 and 3.21, Spillane 2014). To that end, define the projection operators

$$\Pi^j_{m_j} v = \sum_{k=1}^{m_j} a_{\Omega_j}(\Xi_j v, \Xi_j p^j_k) p^j_k$$

where $v \in V_h$ and p^j_k is the k-th eigenvector from subdomain j. Note that we work with the modified GenEO problem from above. Further define the decomposition $v = \sum_{j=0}^{N} z_j$ according to Definition 30 with the particular choice

$$z_0 = \Xi_j \Pi^j_{m_j} v|_{\Omega_j}$$

and

$$z_j = \Xi_j (v|_{\Omega_j} - \Pi^j_{m_j}) v|_{\Omega_j}, \qquad 1 \leq j \leq N.$$

Using the approximation property from Lemma 8, this decomposition can be shown to form a localized splitting with

$$\|z_j\|_{a,\Omega_j} \leqslant \sigma^j_{m_j+1} |v|_{a,\Omega_j}.$$

Proof: Remains as before.

As before, the final condition estimate can be obtained from Theorems 8 and 9, yielding the same result as the original GenEO analysis.

[1] P. Bastian, R. Scheichl, L. Seelinger and A. Strehlow, Multilevel Spectral Domain Decomposition, accepted for publication in SISC, 2021