https://github.com/Amberisfree/UQ?tab=readme-ov-file

Acceleration of Uncertainty Quantification for Reservoir Simulation on Parallel System

Student Name: Amber Chang

Supervisor Name: Anne Reinarz

Submitted as part of the degree of M.Sc. MISCADA to the

Board of Examiners in the Department of Computer Sciences, Durham University

2024

Abstract —

Uncertainty Quantification (UQ) is central to understanding the nuance of complex numerical problems of real-world phenomena, and statistical impacts of computational problems related to high dimension. While most reservoir engineering applications involve time-dependent simulation of stochastic process, they tend not to take uncertainty of input parameters into the construction of geophysical models. Instead, they incorporate uncertain factors in limited ways, assessing the uncertainty of models in isolation, therefore falling short of capturing the complexity of parameter-model interactions. In this paper, we discuss and operationalize a definition of an emerging category of model evaluation, which focuses on the assessment of input-output or the process and the outcomes of model predictions. First, we argue that UQ can be used to increase the validity of statistical evaluations, assess direct sensitivity impact, and guide future assessments of models' prediction impact. Second, we propose an variance-reduction—focused UQ design framework – containing an input-output interaction taxonomy – with three stages: (1) identifying the probability distribution of input parameters , (2) improving the sampling techniques to accelerate simulation, and (3) choosing strong scaling point for parallel computing on High Performance Computing (HPC). Third, we apply our framework to two potential evaluations for probabilistic and computational complexity analysis. Finally, we conclude with tangible recommendations for addressing concerns over costs, confidence intervals, and efficiency of UQ application.

Keywords — Uncertainty Quantification, Quasi-Monte Carlo simulation, scientific machine learning, parallel computing, carbon capture

I INTRODUCTION

Uncertainty quantification plays a pivotal role in both traditional simulation and state-of-the-art machine learning models. **One significant advantage of UQ** is its capacity to address uncertainties that arise from various sources, such as parameter variability, structural model inadequacy, algorithmic errors, experimental measurements, and interpolation challenges [1]. In scientific research, UQ helps researchers assess the reliability of their computational models and simulations, particularly in fields like material science, engineering, and computational finance. By incorporating UQ methodologies, scientists can evaluate the impact of uncertainties on their predictions, identify areas of high uncertainty, and make informed decisions based on quantified risk levels [27].

To study real-world phenomena in which systems evolve randomly, we therefore need probabilistic model rather than deterministic ones. Such systems are usually studied as a function of time and their math model are called stochastic models [14]. Computing has advanced scientific understanding by transitioning from model-based simulations to data-centric approaches. However, **uncertainty** poses a significant challenge to developing predictive physics-based models. This uncertainty arises from inputs that characterize the state of the system, and a key task is to understand the impact of these **uncertain inputs on the output of interest**, a process known as sensitivity analysis [12].

Climate modelling or other earth science applications have incorporated their geophysical process to time-dependent simulations of partial differential equations (PDEs). These deterministic models map a given set of parameters to specific predicted values. However, in many applications, **this approach is insufficient**. Uncertainty in model parameters, arising from limited knowledge or measurement accuracy, can significantly impact the accuracy of predictions. This uncertainty needs to be accounted for to avoid overlooking potentially dangerous scenarios with potentially fatal consequences in real-world applications[36]. To elaborate, parameter uncertainty is often represented using probability distributions. The process of determining how these stochastic parameter distributions affect the distribution of model predictions is known as a forward uncertainty quantification (UQ) problem[36].

Recognizing these limitations, there has been an increasing recognition of the need to address how simulation incorporates uncertain factors. In this work, we motivate the need for uncertainty in forward problem context within a framework to design input-output interactions. To demonstrate the practicality of our method in practical applications related to forward problems, we use a Fortrain model of reservoir engineering as our case study. This model is developed by OpenGoSim, one of the software developing companies, who has developed a simulation of carbon storage for 200 years by using a scientific modelling tool with a view to achieving the net-zero emission target. However, to study the outcome of uncertainty from their simulated input and output data, it requires us to sample with tones of simulated data according to Central Limit Theorem. Therefore, it is crucial that we have smart sampling algorithms to tackle the uncertainty quantification at a good convergence rate.

In this project, we investigate some advanced randomized sampling techniques from UQ framework which not only accurately quantify the uncertainty of parameters of the existing simulation model developed by OpenGoSim, but also significantly improve the computational efficiency of the traditional Monte Carlo method. One of the golden approaches that has been widely discussed in literature and have been proven to outperform traditional sampling methods in terms of reducing variance and increasing computational efficiency is Quasi-Monte Carlo (QMC). Throughout the whole project, the aim of using an Uncertainty Quantification framework is to achieve improved efficiency and accuracy in simulation tasks of carbon storage, particularly in scenarios involving high-dimensional integration or complex stochastic models. By implementing Quasi-Monte Carlo to our model, we aim to tackle complex computational challenges more effectively and achieve higher levels of accuracy and reliability in their numerical analysis and simulation tasks, demonstrating the strength of uncertainty quantification and probabilistic sampling strategy. **Our key contributions are:**

- **Defining input parameters and clarifying how they contribute to our understanding of model uncertainty.** We use an Uncertainty Quantification framework to achieve improved efficiency and accuracy in simulation tasks of carbon storage
- **Developing an accurate and efficient evaluation design framework and Instantiating the framework.** We also evaluate this randomized algorithm in terms of stability, convergence, and accuracy. We simulate the process of CO2 saturation by solving the Darcy Equation on the HPC system. We leverage UM-Bridge as an interface between our model and the UQ framework.
- **Acceleration of the algorithms on HPC by exploiting multi cores** We accelerate the process of UQ through both sample parallelization and MPI parallelization and identify optimal number of CPUs for different settings of stochastic process.
- **Providing an Statistical Analysis** We have done a thorough analysis of how input parameters contribute to the runtime.

We simulate the process of the CO2 saturation by solving the Darcy Equation on HPC system. We leverage UM-Bridge as an interface between our model and the UQ framework [35]. This involves wrapping the model with UM-Bridge within a client-server architecture, allowing us to conduct a deeper analysis of uncertainty propagation using QMCpy. We also evaluate this randomized algorithm in terms of stability, convergence, and accuracy. We further show parallel scalability of our UQ algorithm and implementation via Amdahl's law.

II BACKGROUND AND RELATED WORKS

Randomness and probabilistic methods play a key role in modern computer science. We first clarify the key terms used in this paper. Then, we review research on the evaluation of Uncertainty Quantification in the context of probabilistic computing. Finally, we propose using Quasi-Monte Carlo to solve our problem.

A Defining Uncertainty Quantification As a Probabilistic Estimation Problem

We follow existing work in adopting a wide definition of uncertainty quantification, which encompasses aleatoric and epistemic uncertainty. Aleatoric (aka statistical) uncertainty refers to inherent randomness in a process, such as coin flipping, where even the best model can only provide probabilities for outcomes. Epistemic (aka systematic) uncertainty stems from a lack of knowledge about the best model, reflecting the decision maker's ignorance [23]. Unlike aleatoric uncertainty, epistemic uncertainty can potentially be reduced with additional information.

In Uncertainty Quantification (UQ), much of the research on model evaluation has focused on analysing and improving domains like uncertainty propagation, sensitivity analysis, statistical inference and model calibration (cite). Additionally, other research uses various sampling strategies from optimal experimental design with model validation to improve decision making under uncertainty (cite). Recent research has highlighted that many models tend to overestimate their confidence in probability predictions, potentially leading to confident but incorrect predictions [19]. This **overconfidence** arises because the predicted probabilities may not accurately reflect the true likelihood. Epistemic uncertainty is a key factor contributing to **overconfidence** in model predictions. Calibration issues often arise from neglecting this form of uncertainty. Bayesian Model Averaging (BMA) offers a robust approach to mitigate **overconfidence** by explicitly accounting for epistemic uncertainty [40].

Our goal is to accelerate the process of using one set of samples to detect a set of ranges or estimate the probabilities of a set of ranges, where the set of possible ranges is large to achieve desirable convergent rates.

B Sampler

Recent work on the importance of common sampling techniques includes approximation theory, error estimation, stochastic modeling, and Monte Carlo methods. This is important because sampling enables us to represent the inherent uncertainty in the system being modeled. Many optimization problems involve optimizing black-box functions, where evaluating a sample requires physical experimentation or time-consuming simulation, potentially taking weeks or months per experiment [38]. This limits the total number of evaluations possible. Numerous works have proposed a variety of both Bayesian and non-Bayesian methods to model the heteroscedasticity introduced by the stochastic data generation process for predicting the uncertainty estimates along with a model's predictions [34].

Monte Carlo (MC) and Bayesian Optimization (BO) are both powerful uncertainty quantification techniques, but they excel in different scenarios. However, recent research on **non-BO techniques has been relatively limited**, with some exceptions. If the objective function is relatively simple and well-behaved, MC provides a good choice. MC can efficiently explore the search space and find good solutions, especially when the function is smooth and unimodal and computational cost is a major concern [3]. MC is often computationally cheaper than Bayesian optimization, especially when the number of evaluations is large. In summary, MC is a good choice for simple, well-behaved functions with low evaluation costs, while Bayesian optimization is better suited for complex, expensive functions with many local optima. The choice between the two depends on the specific characteristics of the optimization problem.

In our application, we employ Monte Carlo (MC) based sampling techniques to solve partial differential equations (PDEs). This approach has a rich history in computational mathematics. Marschner provides a comprehensive review of the historical use of Monte Carlo methods for PDE solving [28]. The roots of this approach can be traced back to the mid-20th century, with pivotal contributions from two seminal works: the 1967 paper by Courant, Friedrichs, and Lewy [9], and the earlier 1950 study by Forsythe and Leibler [13].

Monte Carlo (MC) sampling techniques are primarily motivated by the difficulty of analytical integration in complex systems. These numerical methods offer a practical approach to evaluating integrals when closed-form solutions are unavailable or computationally infeasible. One key application of MC sampling is in tracking errors generated by stochastic processes, which are often intractable through conventional computational methods. Monte Carlo methods act as unbiased estimators, offering approximately correct answers that converge to the true value as the sample size grows. The key advantages of Monte Carlo techniques are their simplicity and robustness. They excel at evaluating black box functions through sampling, perform effectively in high-dimensional spaces, and can handle discontinuities in the target function. These properties make Monte Carlo methods versatile and widely applicable across various computational domains.

While Monte Carlo methods guarantee unbiased estimates, they often suffer from high variance and can be computationally expensive, especially when integrating over large, high-dimensional spaces. To address these issues, we consider Quasi-Monte Carlo (QMC) methods, which offer significantly improved error performance. QMC achieves higher quality estimates with fewer samples and faster convergence rates. The key innovation behind QMC is the use of deterministic, low-discrepancy sequences instead of random sampling. These sequences are designed to be well-distributed across the sample space, appearing random while actually being precisely calculated. The fundamental principle of QMC is to eliminate randomness in favor of carefully constructed sample points that provide more uniform coverage of the integration domain. This approach allows

QMC to explore the target function more efficiently, resulting in improved accuracy and reduced computational cost compared to standard Monte Carlo methods.

C Reviewing Efficient Solution of Forward Problem via Quasi-Monte Carlo

Kroese evaluates several variance reduction techniques, which aim to reduce the variance of Monte Carlo estimates, leading to more efficient simulations and faster convergence to accurate results [25, 376-379]. In particular**, quasi random number generation,** unlike uniform random number generator, exploits deterministic choice of points, thus exhibiting more regularity and uniformity in the evaluation of multidimensional integrals. This means that discrepancy of simulated data from QMC simulation is less obvious and the error bound decreases at a better rate than $\mathcal{O}(N^{-1/2})$. In Hung's review paper of QMC, he highlights the concept of "space-filling" design but points out a typical quasi-Monte Carlo sampling algorithm's weakness of requiring a huge amount of computational cost in high dimensional sampling settings [24]. To tackle this aforementioned computational difficulty, two easy-to-implement algorithms, such as optimized Latin hypercube sampling (OLHS) and *randomized quasi-Monte Carlo method*, are introduced along with two accelerators (i.e., augmentation of low-dimensional embedding and parallel computing). In sum, it is worth noting that these two proposed sampling schemes can be improved to achieve $\mathcal{O}(N^{-1})$ convergence rate [26, 33]. Furthermore, instead of choosing good low-discrepancy points, new direction of the research focused on establishing the universality of Quasi-Monte Carlo (QMC) integration methods by extending previous results and considering two classes of QMC point sets (linearly scrambled digital nets and polynomial lattice point sets) across three weighted function spaces with varying smoothness classes. It addresses the absence of universality in terms of weights in the existing literature and introduces new results on the dual properties of linearly scrambled digital nets and polynomial lattice point sets [17].

Similar to QMC, **multilelvel Monte Carlo** is also widely researched to improve crude Monte Carlo. Mike Giles aims to highlight the fundamental simplicity of the multilevel method, which essentially operates as a recursive control variate strategy. It leverages inexpensive and imprecise approximations of a random output variable as control variates to enhance the accuracy of more costly but precise approximations [15]. Having researched a few applications related to this method, we found large-scale stochastic subsurface simulations are developed with improved multilevel Monte Carlo (MLMC) method for estimating cumulative distribution functions (CDFs) of a quantity of interest [29]. In this paper, a widely used approach for describing uncertainty in subsurface flow is to represent porous media property (eg. permeability) as a parameterized random field. Another example is uncertainty quantification for groundwater flow [8]. Finally, Giles et al elaborate on the use of a randomized quasi-Monte Carlo estimator on each of the levels can specifically improve the asymptotic order of convergence in the MLMC method [16]. These findings have led us to study the future development of MLQMC.

The integration of QMC and MLMC techniques offers a compelling direction for advancing research in uncertainty quantification. A new application of **MLQMC** is proposed to the case of direct KL decomposition for log-Gaussian random fields [10]. This method is a consequence of combining the MLMC algorithm with a randomized QMC estimator instead of the MC estimator. In this paper, they use the MLQMC algorithm developed by Mike Giles [16]. In summary, the integration of quasi-Monte Carlo methods into Multilevel Monte Carlo techniques can bring about several advantages, including accelerated convergence, improved efficiency in high-dimensional settings, and enhanced robustness in numerical simulations. By combining the strengths of QMC and MLMC, we aim to tackle complex computational challenges more effectively and achieve higher levels of accuracy and reliability in their numerical analysis and simulation tasks.

During the research, understanding and quantifying the uncertainties associated with subsurface properties is crucial for making informed decisions in subsurface environmental management, especially as we strive to achieve net-zero targets. The accuracy of predictions about fluid behavior, which directly impacts decisions related to carbon capture and storage, geothermal energy, and groundwater management, hinges on accurately accounting for these uncertainties. To investigate uncertainty propagation using Quasi-Monte Carlo methods (QMCpy) [7], we leverage UM-Bridge as an interface between our model and the UQ framework [35]. This involves wrapping the model with UM-Bridge within a client-server architecture, allowing us to conduct a deeper analysis of uncertainty propagation using QMCpy.

III UQ WORKFLOW: NOTATION AND SETUP

In physical sciences, experiments frequently aim to determine the value of a specific quantity. However, due to experimental uncertainties, there is always an associated error in the measured value. These uncertainty sources will be explored later in this project. While we can never pinpoint the exact value with absolute certainty, error analysis or uncertainty quantification allows us to define a range within which we believe the true value lies, with a specified level of confidence. This approach provides a framework for understanding the limitations and reliability of our experimental results [22]. **We begin by outlining the uncertainty quantification (UQ) process. Next, we will delve into the structure of UM-Bridge. In the next section, we demonstrate how this UQ framework is applied to our geophysical application, OpenGOSIM.**

A Statistical Analysis of Simulation Data

Monte Carlo simulation refers to a simple algorithm for computing numerical quantities (eg. expectation and probability distribution) and utilizes computer to implement random experimentation. In contrast, simulation models initiate the behaviour of real-time systems, encompassing a collection of interacting entities or objects forming a complex whole. In this project, our goal is to use the UQ framework to study the uncertainty which exists in simulation models, either as a form of systematic or random errors.

A.1 UQ Flowchart

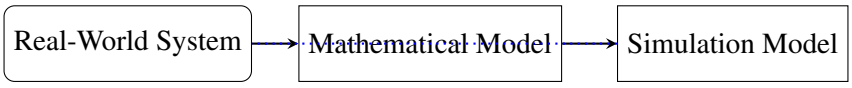


Figure 1: UQ Analytical Flowchart

The UQ (Uncertainty Quantification) analytical flowchart provides a comprehensive overview of the UQ framework. This process begins with the observation of a real-world system, from which we derive knowledge about the world. We then incorporate this knowledge into a mathematical model, which serves as the foundation for developing a computer simulation model. This simulation model is used to analyze uncertainties and quantify the reliability of its predictions. The process is iterative, allowing for continuous refinement and improvement of the model based on the results of uncertainty analysis and validation against real-world data.

The simulation data is reproducible because the simulation model is completely known; however, the data produced by simulation experiments can be viewed as outcome of random variables, random vectors, or stochastic process. Therefore, the objective of the simulation is to draw conclusions about various characteristics of these random objects, such as their expectations, correlations, and distributions. We also have some underlying assumptions about the data, $\theta_1, \ldots, \theta_N \overset{\text{iid}}{\sim} f$, form an i.i.d sample from some distribution, and we need to estimate the performance measure $\mu = E(\mathbf{Y})$ in which $\mathbf{Y} = F(\mathbf{X}; \theta)$ through Monte Carlo sampling and Integration with $Y \overset{\text{iid}}{\sim} f \quad Y = \{Y_1, \ldots, Y_n\}$.

B Notation Setup

IV MONTE CARLO SIMULATION METHODS

This section can present the proposed solutions of the problems in detail. The design and description of the implementation details should all be placed in this section. You may create several subsections, each focusing on one issue.

A Randomized Algorithm and The Monte Carlo Method

The Monte Carlo method refers to a collection of tools for estimating values through sampling and simulation. These techniques are ubiquitous in almost all areas of physical sciences. Estimating through sampling is

Table 1: UNITS FOR MAGNETIC PROPERTIES

Symbol	Meaning
$\mathbb{X}$	input value (deterministic)
θ	input parameter which influence input and have influence on output (probabilistic)
$\mathbb{Y}$	output value
π	distribution of input parameter
$\mathbb{F}$	Function, Data generation process, mapping from input to output
μ	population mean of Quantity of Interest
$\hat{\mu}$	approximated value of Quantity of Interest

often more complex than using a simple experiment to estimate the value of the constant π.

Data: 1. Generate $Y_1, \ldots, Y_N \overset{\text{iid}}{\sim} f$ (for example, from independent simulation runs).
Result: 2. Calculate the point estimate $\bar{Y}$ and confidence interval of $\mu = \mathbb{E}Y$.
initialization It is often the case that the output Y is a function of some underlying random vector or stochastic process; that is,

$$Y = F(\mathbf{X}; \theta),$$

where F is a real-valued performance function and θ is a random vector or process. Provided that independent copies of Y can be generated in finite time, Algorithm 1 can be used for both static simulation models (in which case θ is a random vector) and dynamic models (in which case θ represents a time-dependent stochastic process).

Algorithm 1: Crude Monte Carlo for Independent Data

B Crude Monte Carlo simulation

The Monte Carlo method stands out for its sampling advantages, especially in simulations. It offers a more efficient way to calculate the quantities we're interested in. Unlike real-world data, simulation data is perfectly reproducible because the underlying model is fully understood. This means we can generate as much data as we need, limited only by the time we are willing to take to run the simulation and the storage capacity of our computer. Furthermore, the data produced by simulations can be interpreted in various mathematical frameworks, such as random variables, vectors, time series, or stochastic processes, providing a powerful tool for analyzing complex systems [25, 301-302]. In short, Monte Carlo sampling provides a comprehensive framework for statistical analysis of simulation data.

We study numerical integration for functions defined over s-dimensional unit cube. The quantify of interest(QoI) can be computed through Monte Carlo simulation, for any real function $f : [0, 1]^s \to \mathbb{R}$

$$I_s(f) := E[f] = \int_{[0,1]^s} f(\mathbf{x}; \theta)\, d\mathbf{x} \tag{1}$$

The expectation of QoI can be approximated via the sample average

$$\hat{\mu} := \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}; \theta_i) \tag{2}$$

However, Monte Carlo simulation, which uses random number generator, suffers fro m slow convergence, mainly because the sequence of numbers that it generates is independent and uniformly distributed (ie. i,i,d). From such a stream, the standard error of our expected value decreases at the rate $O(N^{-1/2})$.

$$E[\mu - \hat{\mu}]^{1/2} := O(N^{-1/2}) \tag{3}$$

The downside of Crude Monte Carlo mainly lies in its assumption, the data, say $\theta_1, ..., \theta_N$ form an i.i.d sample from some distribution. This Independence of data stream is proven to result in slow convergence by

many literature that when we are estimating the performance of the measure (eg. sample mean and sample variance).The Crude Monte Carlo method may require **a significantly larger number** of samples to achieve accurate estimates.

Several techniques of variance reduction will be discussed in the following sections to improve CMC convergence rate.

C Quasi-Monte Carlo simulation

Quasi-Monte Carlo (QMC) methods estimate the integral of a function, denoted as $I_s[f]$, by averaging function evaluations over a strategically chosen set of points, 'P'. This set 'P' isn't just any collection of points; it is a multiset, meaning it allows for dependence of points, and the number of times each point appears is significant. The size of this multiset, counting repetitions, is represented by 'N', denoted as $N := |P|$.

$$I_s[f] := \int_{[0,1]^s} f(\mathbf{x};\theta)\, d\mathbf{x} = \mu \tag{4}$$

Essentially, QMC methods take advantage of the fact that by carefully selecting the points in 'P', they can improve the accuracy of the integral approximation compared to standard Monte Carlo methods, which randomly sample points.

$$Q_s(f) := \frac{1}{N} \sum_{\theta \in P} f(x;\theta) = \hat{\mu} \tag{5}$$

To elaborate, QMC aims to accelerate the rate of convergence by using deterministic sequences instead of pseudo-random. Quasi-random points are no longer independent, but do have a high degree of uniformity, which is often expressed in terms of their discrepancy. The discrepancy of the sequence is the measure of its uniformity and low discrepancy means less occurrence of spatial clusters and gaps. Given a set of points $P = P_1, ..., P_N$ in a unit hypercube $D = [0,1]^s$, the measure of uniformity is described as:

$$D(P_N) = \sup_{x \in \mathcal{D}} |Q_s(f) - I_s(f)| \tag{6}$$

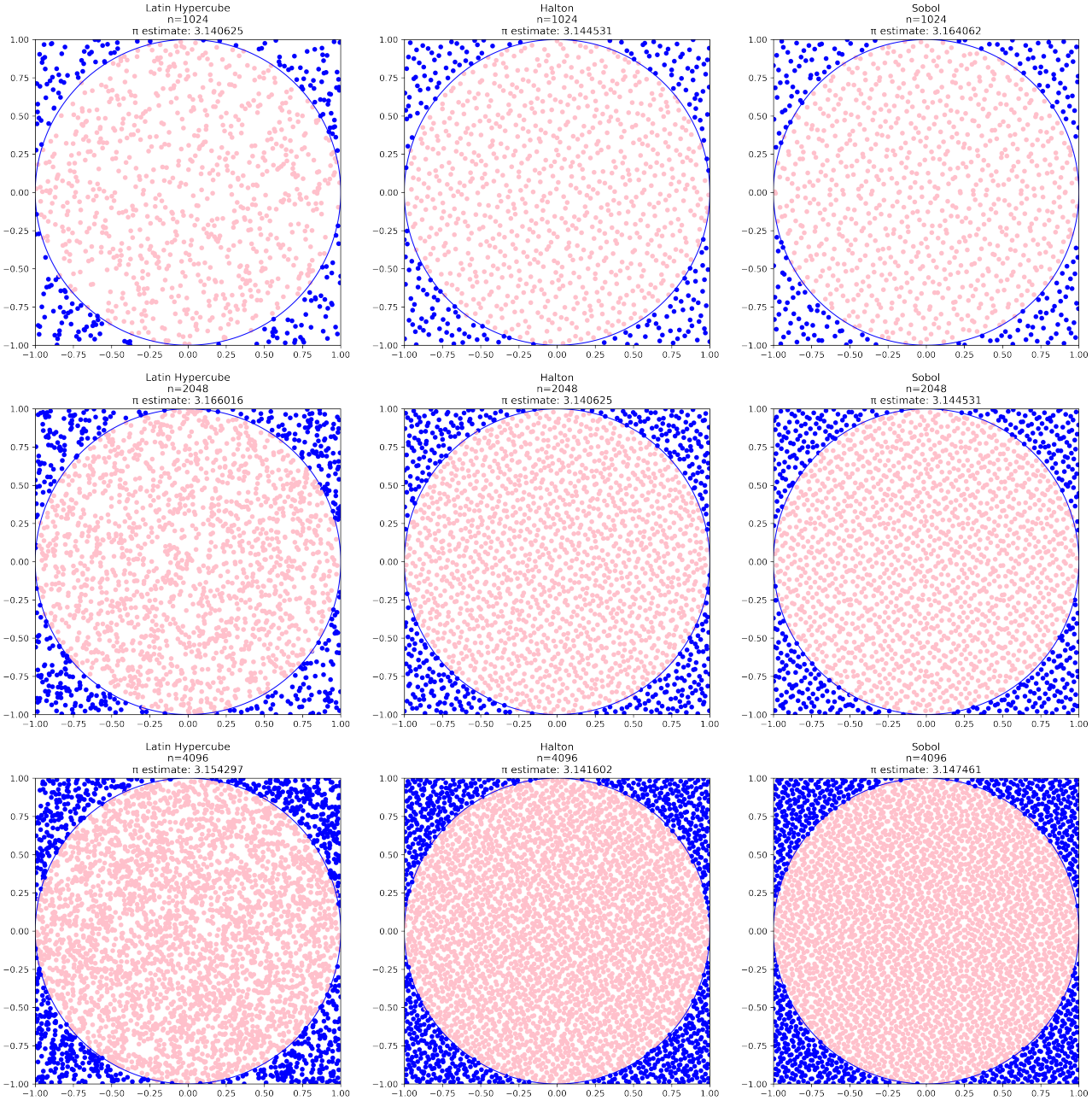
This "space-filling" design results in low discrepancy and hence leads to a more efficient computation of QoI. Hence, the convergence rate is largely increased but is directly linked to the dimension. The standard error of our expected value decays at the rate $O(N^{-1})$.

$$E[\mu - \hat{\mu}]^{1/2} :\approx O(N^{-1}) \tag{7}$$

Table 2: **The comparison of the Monte Carlo (MC), lattice-grid, and quasi-Monte Carlo (quasi-MC) sampling method** [24]

Sampling method	Convergence rate	Sparseness (for a fixed n)	Sampling scheme	Computational complexity
MC	$O(n^{-1/2})$	Not obvious	Random	Medium
Lattice-Grid	$O(n^{-k/d})$	Yes	Deterministic	Low
quasi-MC	$O(n^{-1/\epsilon})$	Improved	Deterministic	High
Multilevel-MC	$O(n^{-1/2-\alpha})$	Improved	Random	High

C.1 Space-Filling Design



D Improved QMC: Accelerator : Parallel and Distributed computing

Furthermore, to increase the convergence rate, we will be experimenting two QMC estimators scrambled sobol on HPC using parallelization techniques to accelarate the whole process of the integration.

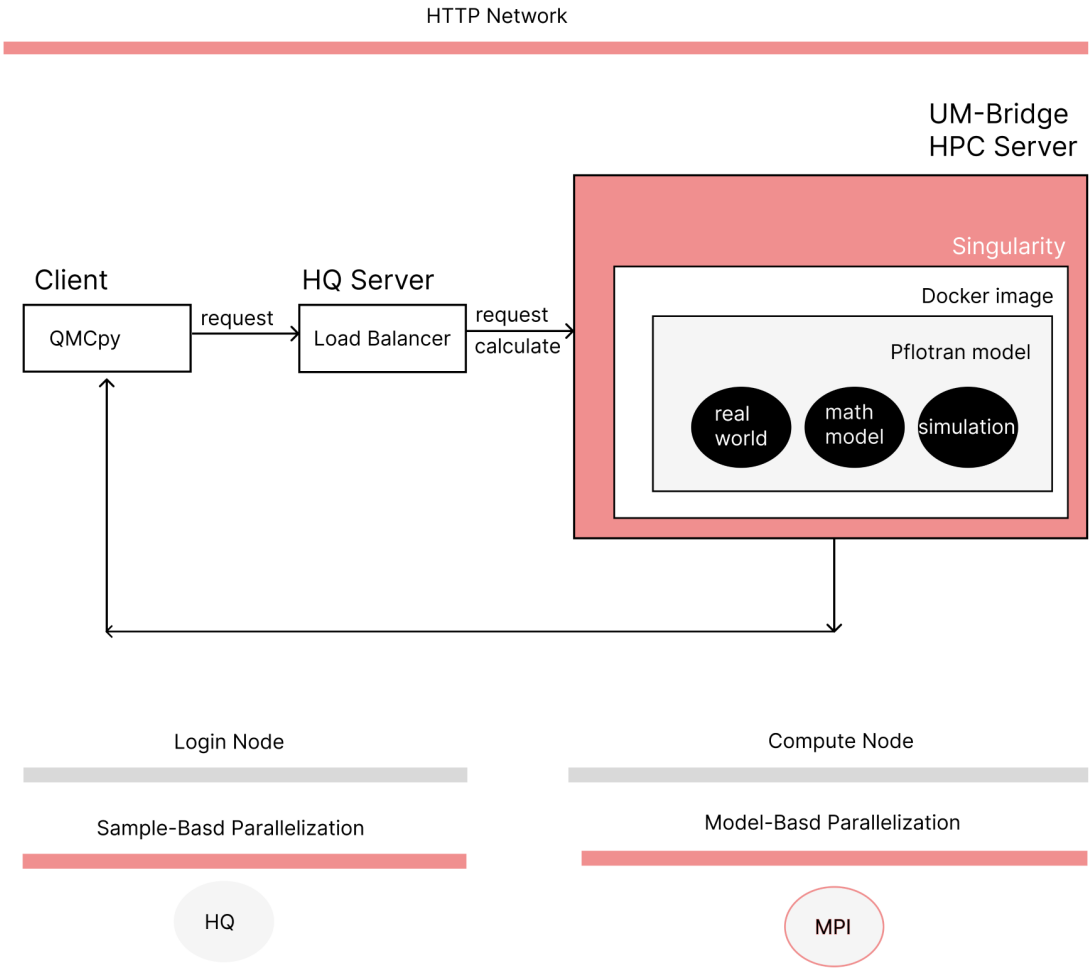
1. Randomized QMC
2. Parallel Sampling

V UM-BRIDGE ARCHITECTURE

We intend to incorporate the Monte Carlo statistical simulation process with UM-Bridge Architecture to capture the uncertainty of a deterministic model. Later in this section, we will explain how this design will overcome the challenge presented by the deterministic model's overconfidence prediction. Furthermore, we

will elaborate on how this design enables our further study of uncertainty propagated through physic-based simulation.

In their work "Democratizing Uncertainty Quantification," Seelinger, Reinarz, et al. present UM-Bridge, an emerging technology designed to facilitate more robust and extensive statistical analysis. This approach addresses the critical need for comprehensive risk assessment in safety-sensitive applications, where understanding potential uncertainties is critical.



A *Separation of concern*

Their approach to uncertainty quantification (UQ) analysis prioritizes separation of concerns, decoupling the statistical model from the expert's physical simulation of any typical deterministic model. This design frees UQ models from compute-intensive calculations that typically require low-level languages and efficient numerical solvers. Additionally, it ensures the reproducibility of statistical results.

B *Forward UQ*

Forward Uncertainty Quantification (UQ) is a data generation process that involves a forward operator $F(\theta)$. This operator takes random parameter input vectors in n dimension, typically defined by a known probability distribution $\pi(\theta)$. The primary goal of forward UQ is to determine the unknown distribution of the output $F(\theta)$ vector in m dimension. This process enables the solution of various problems through data generation, such as numerical integration.

Mathematically, we can express this as:

$$\theta \sim \pi(\theta) \tag{8}$$

$$\underbrace{\theta}_{\text{uncertainty}} \rightarrow \mathbb{F} \qquad \rightarrow \qquad \underbrace{F(\theta)}_{\text{inference probability distribution of the output}} \tag{9}$$

$$F : \mathbb{R}^n \rightarrow \mathbb{R}^m \tag{10}$$

where Y represents the model output, and we aim to characterize its distribution given the uncertainty in θ.

C Parallel Backbone

The primary design goal of UM-Bridge's network-based architecture is to encapsulate physical simulations within containers. This approach allows for deployment on High-Performance Computing (HPC) systems, enabling the scale-up of Uncertainty Quantification (UQ) applications on parallel compute clusters. The server and client communicate via the HTTP protocol, offering two layers of acceleration benefits:

1. **MPI** (Message Passing Interface) allows Partial Differential Equation (PDE) solvers to utilize HPC resources effectively.
2. **HQ** (Hyper Queue) enables a cluster-side load balancer to concurrently handle requests from multiple clients' evaluations.

This design enhances resource utilization efficiency and boosts performance for complex Uncertainty Quantification (UQ) applications by implementing multiple sample generation and task execution in parallel. The Message Passing Interface (MPI) enables each task to utilize multiple cores, while Hyper Queue (HQ) allows multiple samples to run concurrently within a single node across multiple cores. This multi-level parallelism effectively exploits the advantages of embarrassingly parallel scenarios.

An MPI-based parallel computing architecture employing a hierarchical master-slave model with distributed slave partitions. The system consists of a central master node overseeing multiple groups of slave processors. Each slave partition operates autonomously, managing a subset of the overall workload. The master node coordinates task distribution, load balancing, and result aggregation across the partitioned slave groups [2].

HQ is a task scheduling runtime designed to efficiently execute a large number of "embarrassingly parallel tasks" on *Slurm* cluster [5]. HQ's design of scheduling facilitates higher throughput via "work stealing" to achieve better speed-up. This scheduling design allow HQ workers in compute node to have evenly-distributed workloads assigned by the HQ server, not through effective communication between processors but through work stealing, further preventing overhead issues and promoting load balancing. In essence, the key contribution in parallelization of HQ lies in the design of its architecture. The architecture is composed of three main components, in which *Server* coordinates workers and handles tasks to load balance the cluster, *Worker* is a process which execute tasks submitted by the client, and *Client* is the user-facing API used to run distributed computations [6].

The interaction of client, server, and workers inside the architecture is the key to "embarrassingly parallel." To elaborate, an embarrassingly parallel problem in computing refers to a task that can be effortlessly divided into multiple independent sub tasks for parallel processing. According to the author of *the art of multiprocessor programming* , this type of workload, also known as perfectly parallel, delightfully parallel, or pleasingly parallel, requires minimal or no communication between the parallel tasks or their results. In addition, the design of stealing algorithm for scheduling allows idle thread to steal thread from others; if all threads are always busy, then they do not waste time offloading work on another [20, 381]. This "work stealing" design additionally contributes to the success of load balancing. In summary, HQ combines the lack of necessity of communication between processors/workers but with great design of work stealing to achieve scaling through task aggregation and load balancing.

D Uncertainty Propagation Benchmark

θ is considered as a random input vector following some unknown distribution $\pi(\theta)$. The goal is to determine properties of the distribution of $F(\theta)$, where F is the forward operator. Therefore, the benchmark's objective is to quantify the effect of uncertainties in model parameters on the model output.

VI APPLICATION: OPENGO SIMULATION

Building on our detailed review of recent work, we propose a novel taxonomy to characterize the core technical challenges of Uncertainty Quantification (UQ). This approach combines QMCpy, our primary Monte Carlo statistical technique, with the UM-Bridge architecture to conduct effective UQ analysis. In this section, we present comprehensive experiments designed to evaluate the performance of uni-modal uncertainty quantification based on UM-Bridge. Our focus is on investigating the uncertainty associated with subsurface flow models developed using OPENGO simulation, considering both systematic and random errors. We begin by briefly explaining Darcy's Law, which governs subsurface equations, and how this Partial Differential Equation (PDE) relates to our forward model design. Following this, we address several key challenges. We begin by identifying and categorizing sources of error in our models, which is crucial for understanding the origins of uncertainty. Next, we focus on developing probabilistic representations of error bounds, allowing for a more nuanced understanding of the uncertainty range. Finally, we tackle the issues of sample complexity in our analysis, ensuring that our results are statistically robust and computationally feasible.

These experiments aim to provide a holistic understanding of the uncertainties inherent in subsurface flow modeling and to demonstrate the efficacy of our proposed UQ methodology.

A Experiment Setup: Forward Model

Subsurface flow models, which simulate the movement of fluids like oil, gas, and water through porous materials, are heavily influenced by the properties of the porous medium itself. These properties, such as permeability and porosity, directly impact model outputs like mass flow rate and pressure. However, due to limitations in measurement techniques or data availability, many of these porous media properties are inherently uncertain. This uncertainty arises from incomplete or inaccurate measurements [21] [37], leading to potential errors in model predictions. **The main aim of our project** is to simulate the process of CO2 injection through a porous medium. Darcy's Law governs the flow equation.

The Darcy velocity $\mathbf{q}_\beta$ for a phase β is given by [32]:

$$\mathbf{q}_\beta = \frac{Kk_\beta}{\mu_\beta}\nabla(P_\beta - \rho_\beta g z) \tag{11}$$

Where K is the saturated media permeability; k_β is the relative permeability of phase β; μ_β is the viscosity of phase β; P_β is the pressure of phase β;ρ_β is the mass density of phase β;g is the gravitational constant;z is the height coordinate.

Our forward model takes parameters (ie. porosity, permeability, and fault opening) as input data. The porosity comes from 12 layers so we have 12 inputs. We define porosity's distribution as Gaussian with mean 0.20 and standard deviation 0.079 based on the source [39].

$$poro := \mathcal{N}(\mu = 0.20, \sigma^2 = 0.79 * 0.79) \tag{12}$$

The dependence of permeability on porosity is specified as

$$perm := 10^{15.6 * poro - 0.9} \tag{13}$$

We are interested in how the errors propagated from our input data to the output. Hence, we develop a framework of uncertainty quantification with three different methods (MLMC, QMC, MLQMC) to compare with the crude Monte Carlo with respect to reducing the variance of our prediction of flow behavior and increasing the computational efficiency of **using partial differential equations (PDEs) with random input parameters.** In the model developed by OpenGoSim, the model problem is **elliptical PDE**s with random coefficients eg. a normal distribution of porosity and permeability derived from porosity based on their correlation. Efficiently generating samples of random fields is a key challenge when solving partial differential equations with random coefficients. Croci's work focuses on efficiently sampling Gaussian fields using quasi-random points within the framework of finite element methods and multilevel quasi Monte Carlo (MLQMC) [11]. Graham proposes a rigorous analysis of methods in construction of Finite Element Approximation in physical space, truncated Karhunen–Loève expansion, and lattice-based quasi-Monte Carlo (QMC) [18].

B Experimental Challenge

The first fundamental challenge in our uncertainty quantification (UQ) approach is to define and distinguish the sources of error in our model in probabilistic terms. This challenge encompasses learning the structural error and decoupling the joint distribution of parameters. To address the complexity of errors residing in our models, we employ different analytical frameworks. We propose a general strategy to combat these issues through two main approaches. First, we separate the probability representation of errors using the Bayesian Model Averaging (BMA) framework through factorization. Second, we learn the statistical associations and dependencies through Infinitesimal Perturbation Analysis (IPA).

B.1 Framing the error in terms of probability

1. **Factorization and Bayesian Model Average**

   Joint distribution of uncertain model parameters is arguably unknown. Insufficient information on parameter inter-dependencies, particularly their correlation or complete joint distribution, poses a significant challenge in precisely quantifying uncertainty in model predictions. This gap in the literature hinders accurate uncertainty estimation [4].

   Often the predictive distribution we want to compute is given by

   $$p(y \mid x, \mathcal{D}) = \int_{\theta} p(y \mid x, \theta) p(\theta \mid \mathcal{D}) d\theta$$

   We combine $\{\boldsymbol{\theta}_{porosity}, \boldsymbol{\theta}_{permeability}\}$ as a set of uncertain model parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_1+d_2}$ with a prior that encodes our prior knowledge about $\boldsymbol{\theta}$.

2. **Infinitestimal Perturbation Analysis (IPA) Analysis**

   A typical setting is where μ is the expected output of a random variable Y whose value is dependent on a simulation parameter vector $\boldsymbol{\theta}$, such that

   $$\mu(\boldsymbol{\theta}) = \mathbb{E}Y = \mathbb{E}_{\boldsymbol{\theta}_2} H\left(\mathbf{X}; \boldsymbol{\theta}_1\right) = \int H\left(\mathbf{x}; \boldsymbol{\theta}_1\right) f\left(\mathbf{x}; \boldsymbol{\theta}_2\right) \mathrm{d}\mathbf{x}$$

   where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2), H(\cdot; \boldsymbol{\theta}_1)$ is a sample performance function, and $f(\cdot; \boldsymbol{\theta}_2)$ is a pdf (for the discrete case replace the integral with a sum). In this context we refer to the parameter $\boldsymbol{\theta}_1$ as a structural parameter and $\boldsymbol{\theta}_2$ as a distributional parameter. An estimation problem formulated with only distributional parameters can often be transformed into one with only structural parameters, and vice versa;

   to estimating $\mu(\boldsymbol{\theta})$ it is often relevant to estimate various derivatives (gradients, Hessians, etc.) of $\mu(\boldsymbol{\theta})$.

3. **Sensitivity analysis:** The gradient $\nabla\mu(\boldsymbol{\theta})$ indicates how sensitive the output $\mu(\theta)$ is to small changes in the input parameters $\boldsymbol{\theta}$, and can thus be used to identify its most significant components.

Additionally, we aim to measure the sensitivity of output values to changes in input parameters. Modeling the structure of error in our models involves considering both the input parameters themselves and our prior knowledge of the true measure's distribution. These considerations pose significant challenges to quantifying the accuracy of our forward model. While the risk of statistical error can potentially be mitigated through repeated experiments (i.e., Monte Carlo simulations) to infer expected values and variances, the more challenging aspect of UQ lies in framing structural errors as distributions. Specifically, we focus on two key areas: quantifying the correlation of input parameters with confidence, and incorporating structural sensitivity parameters and multivariate Gaussian probability density functions (PDFs) to estimate derivatives for sensitivity quantification. This approach allows us to develop a more comprehensive understanding of the uncertainties inherent in our subsurface flow models and to refine our UQ methodology accordingly.

QMC methods offer an efficient means to sample high-dimensional parameter spaces.

Monte Carlo's accuracy increases with sample size [31]. Therefore, to get to the approximately correct answer, we need to ask how many samples we need and how great is our probability estimate. Probably Approximation Correct (PAC) Learning provides a framework for analysis of computational learning from examples. This framework characterizes the complexity of a learning problem in terms of the number of examples and computational needed to provide answers that are approximately correct [30].

VII NUMERICAL RESULTS AND EVALUATION

A *Evolution of Stochastic Process*

This section compares the evolution of two Quantities of Interest (QoIs) over time. Figure 2 illustrates the pressure evolution, while Figure 3 depicts CO2 trapping in a non-ideal space. Despite both QoIs being based on similar linear models, they exhibit distinct behaviors. Conducting an initial exploratory analysis would be beneficial before running simulations with thousands of samples, as it could help determine whether we need to standardize the scale of values to achieve desirable integration.

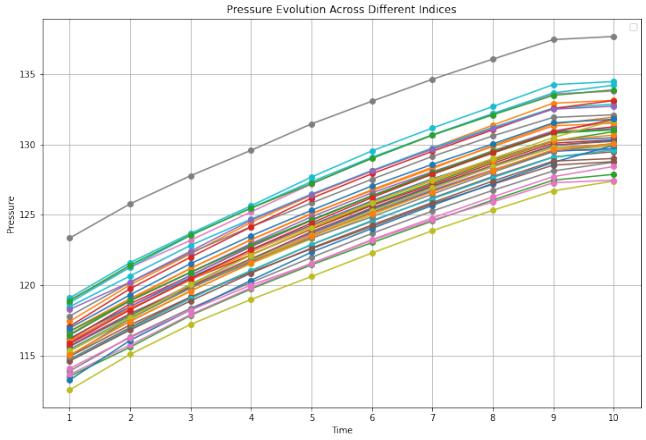


Figure 2: Stochastic Process of Pressure simulated for 1 year using 8 CPU (1 year and 1-year minus 10 timestep)

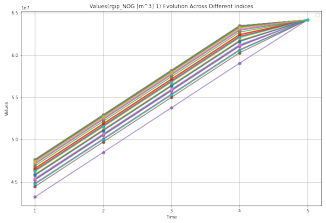


Figure 3: Stochastic Process of No-Go Area simulated for 1 year using 8 CPU (1 year and 1-year minus 5 time steps)

B *Convergence and Error Bounds*

This section summarizes the simulation setup for input and output parameters in Table 3, and provides a description of convergence criteria and error bounds in Table 4. QMCPy offers comprehensive prior simulation information for the Uncertainty Quantification (UQ) analysis, including the prior distributions of input parameters and the type of deterministic samples used. Upon successful integration, the QMCPy client automatically calculates expected values by sampling thousands or hundreds of times until the specified error tolerance is achieved. This approach ensures that the results meet the desired level of accuracy for the UQ analysis.

Mapping	Dimensions	Description
inputSizes	[12]	permeability across 12 sub-surface layers
outputSizes	[2]	CO2 trapped in non ideal space

Table 3: Input VS Output

Tol	N_samples	Solution	QMC Bounds
0.2	n=512	-	-
0.1	n=1024	64077060.966	[64062334.267 64091787.664]
0.05	n=2048	-	-

Table 4: Convergence and Error Bounds for Different Tolerances

Model Information

```
Model input size is 12
Sampling distributions are DigitalNetB2 (DiscreteDistribution Object)
    d               12
    dvec            [ 0 1 2 ... 9 10 11]
    randomize       LMS_DS
    graycode        0
    entropy         252245037573873721345962985657083068403
    spawn_key       ()
and
Gaussian (TrueMeasure Object)
    mean            0.200
    covariance      0.006
    decomp_type     PCA
```

Data Summary

LDTransformData (AccumulateData Object)

- **solution** $[64077060.966, 64077052.294]$
- **comb_bound_low** $[64062334.267, 64062324.943]$
- **comb_bound_high** $[64091787.664, 64091779.645]$
- **comb_flags** [False, False]
- **n_total** 2^{10}
- **n** $[1024, 1024]$
- **time_integrate** 11285.601

CubQMCNetG (StoppingCriterion Object) **abs_tol** 0.100

- **rel_tol** 0
- **n_init** 2^8
- **n_max** 2^{10}

UMBridgeWrapper (Integrand Object)

Gaussian (TrueMeasure Object) **mean** 0.200

- **covariance** 0.006
- **decomp_type** PCA

DigitalNetB2 (DiscreteDistribution Object) **d** 12

- **dvec** $[0, 1, 2, \ldots, 9, 10, 11]$
- **randomize** LMS_DS
- **graycode** 0
- **entropy** $252245037573873721345962985657083068403$
- **spawn_key** $()$

C *Strong Scaling of Parallel Computing*

This section outlines the optimal CPU usage for various time frame settings, ranging from 1 month and 1 year up to 10 years and 200 years. Understanding these optimal CPU numbers will facilitate further Uncertainty Quantification (UQ) analysis for different Quantities of Interest (QoIs). Table 5 provides a comprehensive overview of timeframes and their corresponding optimal CPU counts, based on a micro-perspective analysis of over 1000 sample runs. In contrast, Table 6 focuses specifically on 200-year simulations for a single job using 128 CPUs, analyzing different combinations of samples and MPI parallelization. Figure 4 illustrates our methodology for determining the optimal number of CPUs for 200-year stochastic process simulations. Additionally, we conducted a strong scaling analysis and speedup analysis to evaluate how this 200-year UQ application of the OpenGOSIM model scales up.

Timeframe	Optimal Number of CPUs	Execution Time for 1000+samples
1 month	4 processes	9m04s
1 year	8 processes	19m25s
10 years	16 processes	1h40m10s
200 years	32 processes	6h01m07s

Table 5: Optimal CPUs for Different Timeframe

Sample Parallelization	MPI Parallelization	Average Execution Time(seconds)
1	128	110.30 ± 212.86
2	64	105.43 ± 185.01
4	32	56.24 ± 95.23
8	16	60.04 ± 98.08
16	8	91.01 ± 114.09
32	4	144.99 ± 168.26

Table 6: Scalability for 200-Year Simulation Based on 128 total CPUs

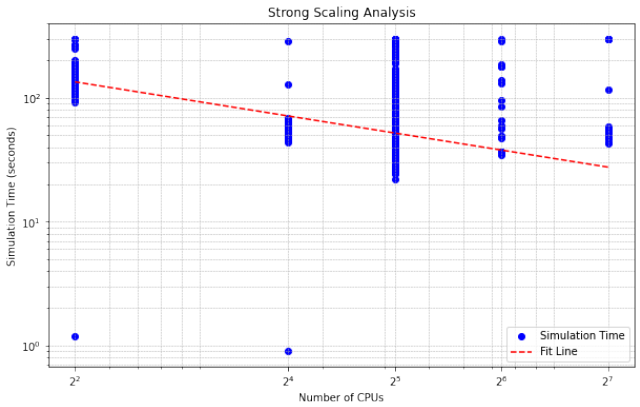


Figure 4: Strong Scaling Analysis

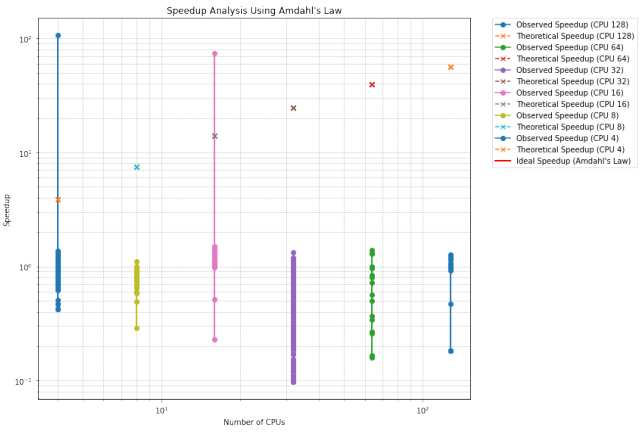


Figure 5: Speedup Analysis with Amdahl's Law, with sequential part as 1/100

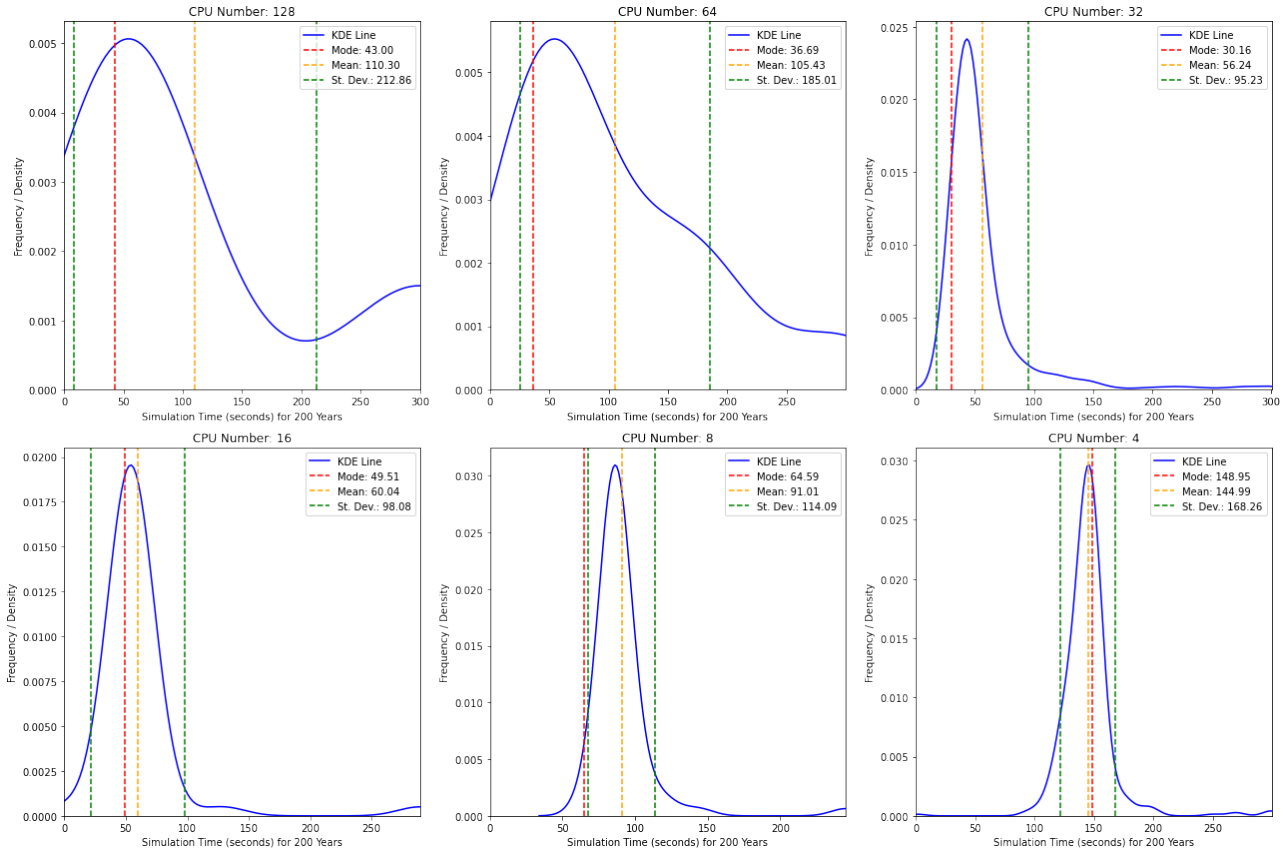


Figure 6: Simulation Time (Per Sample) for 200-Year Simulation Based on 128 total CPUs

D Input Parameters' Contribution to Simulation Time

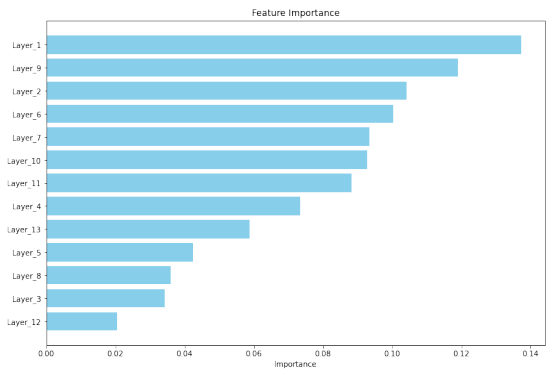


Figure 7: Feature Importance by Regression Tree

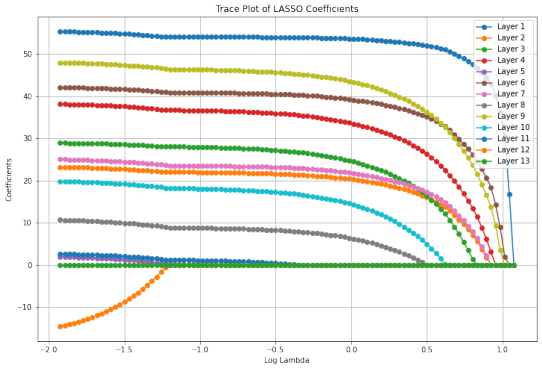


Figure 8: Selection of variables Using Lasso Regularization

To further accelerate our Uncertainty Quantification (UQ) application using the existing technologies of UM-Bridge and HyperQueue on our current hardware resources, we aimed to gain deeper insights into how input parameters influence runtime. Understanding these patterns is advantageous for reducing runtime and building a more robust concurrent system. We applied two machine learning models—a Lasso linear regressor and a decision tree—to analyze feature importance and reduce the parameters to two key features. Surprisingly, both algorithms identified layers 9 and 1 as crucial for runtime performance. The validity of these findings is supported by diagnostic assessments of both regressors using their R-squared values, lending credibility to their results. This discovery opens up opportunities for further investigation to minimize runtime. By focusing

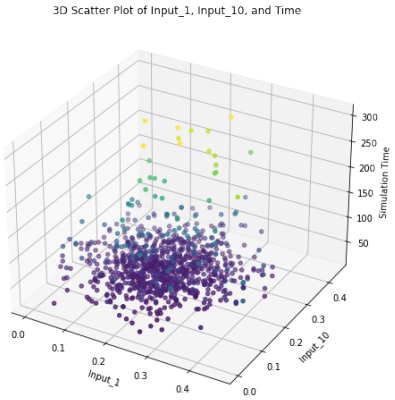


Figure 9: 3D plot for Layer 1 and Layer 9 VS. Simulation Time(Per Sample)

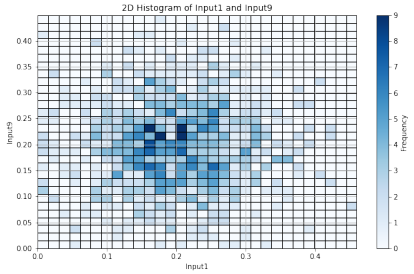


Figure 10: Simulation Density/Frequency for Layer 1 and Layer 9)

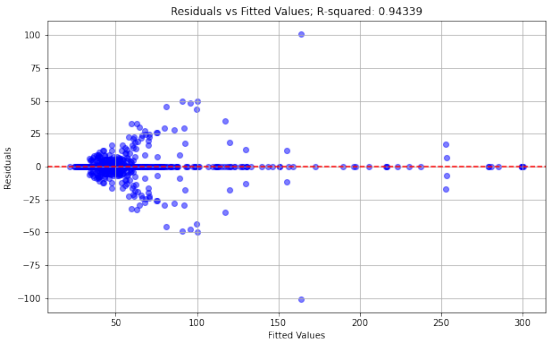


Figure 11: Diagnostics of Input and Time by Tree algorithm

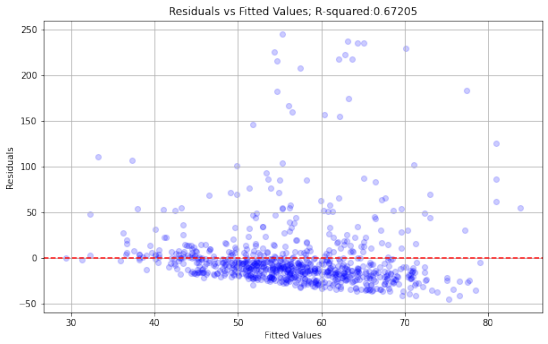


Figure 12: Diagnostics of Input and Time by Lasso algorithm

on these critical layers, we may be able to optimize our UQ application more effectively, potentially leading to significant performance improvements. The consistency between the two different modeling approaches, combined with their statistical validation, provides a solid foundation for future optimization efforts.

E Discussion

Theoretically, we've separated the input parameters' correlation, giving their prior distribution for each of the 12 layers in terms of permeability and porosity. We've successfully instatiated the UQ framework through UM-bridge and conducted a statistical simulation for 1,024 samples using 8 CPUs for 1 year on one node. Although we have tried sample parallelization to accelerate the process, the method needs more investigation to be set up correctly in order to facilitate the acceleration of the whole UQ process. This QMC approach has its own advantages of choosing deterministic points and giving probabilistic values and bounds for the QoI when taking input parameters' uncertainty into account. Especially with UM-bridge and HyperQueue, these network-based statistical and parallelization tools on the HPC have made the development and experimentation process smoother once the setup is complete. Yet, choosing the QoI (Quantity of Interest) is of great importance, as not all applications are feasible for integration through Monte Carlo. This depends largely on the assumption of how the models are run and whether the values they take will converge. We've also compared strong scaling points for different time frames, suggesting that not all timeframes of stochastic processes are suitable for exploiting the same number of CPUs to achieve ideal scaling. This observation is crucial as it significantly affects how we choose the optimal number of CPUs to properly accelerate Monte Carlo simulations and obtain probabilistic, non-overconfident predictions with reasonable error bounds. During experimentation with parallelization of

UQ applications, we have also noted that port issues can result in failed experiments, especially when it comes to massive parallelization through samples where multiple jobs try to occupy the same port. It is therefore reasonable to suspect that... **Limitations:** The QoI's need to be considered; the sample complexity is also crucial in the design of UQ system, as we test our machine. Notice when to stop generating samples can prevent the unneccessary waste of computing resouces and time.

Besides, we have demonstrated how we derive our optimal number of CPUs for parallelization by visualizing 200-year-simulation through different combination of sample parallel and MPI parallel. By conducting strong scaling analysis, we showed that 32 CPUs is the optimal number for this application if the developers intend to do 200 subsurface carbon dioxide storage simulation.

Lastly, we further do the UQ analysis on parallel system with a view to design a low-cost and low-complexity system, we have done a small scientific machine learning experiment for investigating how input parameters will contribute to the runtime. This example is done using the dataset generated and collected from the simulation running for 1 year using 8 CPUs on Hamilton. We observe that to further decrease the runtime, we could focus more our attention to layer 1 and layer 9 in the future.

VIII CONCLUSION AND FURTHER WORKS

Um-Bridge is a framework for benchmarking physic-based simulation in terms of uncertainty quantification. Through a new taxonomy of separation of concerns, quasi-Monte Carlo simulation, and computational power of HPC, UQ enables comprehensive analysis of uni-modal models. In our experiment, we have successfully quantified the uncertainty associated with commercial code of carbon capture of reservoir engineering.
Limitations The evaluation of uni-modal models is done only on a few scientific applications and is limited by the usage of HPC, which is not available to the public who do not have access to the super computers through parallelisation to effectively gain the insights of the experimental result.
Suggestion 1 Further work can improve the quasi-Monte Carlo, such as incorporating Multilevel Monte Carlo or surrogate models, to accelerate the process of predication. Such evaluation is critical to ensure that models are sufficiently robust when deployed in real-world scenarios, to prevent unexpected and unintended consequences. .
Suggestion 2 Future work should also add new metrics, such as adaptive system, to Um-Bridge to measure real-world societal concerns such as robustness and efficiency of models.To more effectively do UQ analysis on HPC, the design of infrastrure, consider integrating the analytics platform with simulation process could be considered to effectively done the UQ analysis on HPC

References

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:376–379, December 2021.

[2] Santiago Badia, Jerrad Hampton, and Javier Principe. A massively parallel implementation of multilevel monte carlo for finite element models. *Mathematics and Computers in Simulation*, 213:18–39, 2023.

[3] Maximilian Balandat, Brian Karrer, Daniel R Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 10628–10639, 2020.

[4] Robert D. Berry, Habib N. Najm, Bert J. Debusschere, Youssef M. Marzouk, and Helgi Adalsteinsson. Data-free inference of the joint distribution of uncertain model parameters. *Journal of Computational Physics*, 231(5):2180–2198, 2012.

[5] Jakub Beránek, Ada Böhm, Gianluca Palermo, Jan Martinovič, and Branislav Jansík. Hyperqueue: Efficient and ergonomic task graphs on hpc clusters. *SoftwareX*, 27:101814, 2024.

[6] Stanislav Böhm and Jakub Beránek. Runtime vs scheduler: Analyzing dask's overheads. In *2020 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*, pages 1–8, 2020.

[7] S.-C. T. Choi, F. J. Hickernell, M. McCourt, and A. Sorokin. QMCPy: A quasi-Monte Carlo Python library, 2020+.

[8] K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup. Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients. *Comput Visual Sci*, 14:3–15, 2011.

[9] Richard Courant, Kurt Friedrichs, and Hans Lewy. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234, 1967.

[10] D. Crevillén-García and H. Power. Multilevel and quasi-Monte Carlo methods for uncertainty quantification in particle travel times through random heterogeneous porous media. *R. Soc. open sci.*, 4:170203, 2017.

[11] Matteo Croci, Michael Giles, and Patrick E. Farrell. Multilevel quasi monte carlo methods for Elliptic PDEs with random field coefficients via fast white noise sampling. *SIAM Journal on Scientific Computing*, 43(4):A2840–A2868, January 2021.

[12] IG Farcaş, G Merlo, and F Jenko. A general framework for quantifying uncertainty at scale. *Communications Engineering*, 1:43, 2022.

[13] George E Forsythe and Richard A Leibler. Matrix inversion by a monte carlo method. *Math. Comp.*, 4(31):127–129, 1950.

[14] Saeed Ghahramani. *Fundamentals of Probability: With Stochastic Processes*. Chapman and Hall/CRC, 3rd edition, 2018.

[15] M.B. Giles. Multilevel Monte Carlo methods. *Acta Numerica*, 24:259–328, 2015.

[16] Michael B. Giles and Benjamin J. Waterhouse. Multilevel quasi-Monte Carlo path simulation. *Radon Series on Computational and Applied Mathematics*, 8:165–182, 2009.

[17] Takashi Goda, Kosuke Suzuki, and Makoto Matsumoto. A universal median quasi-Monte Carlo integration. *SIAM Journal on Numerical Analysis*, 62(1), 2024.

[18] IG Graham, FY Kuo, JA Nichols, IH Sloan, and H Wozniakowski. Quasi-Monte Carlo finite element methods for elliptic PDEs with lognormal random coefficients. *Numerische Mathematik*, 131(2):329–368, 2015.

[19] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[20] Maurice Herlihy and Nir Shavit. *The Art of Multiprocessor Programming, Revised Reprint*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2012.

[21] D. J. Hill and A. G. Tiedeman. Uncertainty in the simulation of groundwater flow and transport. *Advances in Water Resources*, 30(12):3047–3067, 2007.

[22] Ifan G. Hughes and Thomas P. A. Hase. *Measurements and their Uncertainties: A practical guide to modern error analysis*, chapter 2. Oxford University Press, Oxford, 2010.

[23] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110:457–506, 2021.

[24] Ying-Chao Hung. A review of Monte Carlo and quasi-Monte Carlo sampling techniques. *Wiley Interdisciplinary Reviews: Computational Statistics*, Volume Number:Page Numbers, 2023.

[25] DP Kroese, T Taimre, and ZI Botev. *Handbook of Monte Carlo Methods*. John Wiley & Sons, Incorporated, Newark, 2011. Available from: ProQuest Ebook Central. [Accessed 12 June 2024].

[26] Pierre L'Ecuyer. Randomized quasi-Monte Carlo: An introduction for practitioners. In Art B. Owen and Peter W. Glynn, editors, *Monte Carlo and quasi-Monte Carlo methods. MCQMC 2016*, volume 241 of *Springer Proceedings in Mathematics Statistics*. Springer, 2018.

[27] Longze Li, Jiang Chang, Aleksandar Vakanski, et al. Uncertainty quantification in multivariable regression for material property prediction with bayesian neural networks. *Scientific Reports*, 14:10543, 2024.

[28] Zilu Li, Guandao Yang, Xi Deng, Christopher De Sa, Bharath Hariharan, and Steve Marschner. Neural caches for monte carlo partial differential equation solvers. In June Kim, Ming C. Lin, and Bernd Bickel, editors, *SIGGRAPH Asia*, pages 34:1–34:10. ACM, 2023.

[29] Dan Lu, Guannan Zhang, Clayton Webster, and Charlotte Barbier. An improved multilevel Monte Carlo method for estimating probability distribution functions in stochastic oil reservoir simulations. *Water Resources Research*, 2016.

[30] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, USA, 2nd edition, 2017.

[31] Kevin P. Murphy and Francis Bach. *Machine Learning – A Probabilistic Perspective*. MIT Press, September 2012.

[32] OpenGoSim. Mathematical formulation of groundwater flow, 2023. Accessed: 2023-10-26.

[33] Art B. Owen. Monte carlo, quasi-Monte Carlo, and randomized quasi-Monte Carlo. In Harald Niederreiter and Jürgen Spanier, editors, *Monte Carlo and quasi-Monte Carlo methods 1998*, pages 86–97. Springer, 2000.

[34] Utkarsh Sarawgi, Rishab Khincha, Wazeer Zulfikar, Satrajit Ghosh, and Pattie Maes. Uncertainty-aware boosted ensembling in multi-modal settings. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2021.

[35] Linus Seelinger, Anne Reinarz, Mikkel B. Lykkegaard, Amal M. A. Alghamdi, David Aristoff, Wolfgang Bangerth, Jean Bénézech, Matteo Diez, Kurt Frey, John D. Jakeman, Jakob S. Jørgensen, Ki-Tae Kim, Massimiliano Martinelli, Matthew Parno, Riccardo Pellegrini, Noemi Petra, Nicolai A. B. Riis, Katherine Rosenfeld, Andrea Serani, Lorenzo Tamellini, Umberto Villa, Tim J. Dodwell, and Robert Scheichl. Democratizing uncertainty quantification, 2024.

[36] Linus Seelinger, Anne Reinarz, Leonhard Rannabauer, Michael Bader, Peter Bastian, and Robert Scheichl. High performance uncertainty quantification with parallelized multilevel markov chain monte carlo, nov 2021.

[37] D. M. Tartakovsky. Uncertainty quantification in subsurface flow and transport: A review. *Advances in Water Resources*, 51:1–10, 2013.

[38] Yunsheng Tian, Ane Zuniga, Xinwei Zhang, Johannes P. Dürholt, Payel Das, Jie Chen, Wojciech Matusik, and Mina Konaković Luković. Boundary exploration for bayesian optimization with unknown physical constraints, 2024.

[39] AAPG Wiki. Predicting sandstone reservoir porosity. https://wiki.aapg.org/Predicting$_s andstone_r eservoir_p orosity$, 20 $2023 - 10 - 26$.

[40] Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *CoRR*, abs/2002.08791, 2020.

IX APPENDIX

.1 Source of Error

1. **Reducible error:** quantifies the error due to the model's inability to perfectly capture the true relationship between X and Y. This error can be reduced by improving the model's accuracy in approximating the true function.
2. **Irreducible error:** quantifies the inherent noise or randomness in the data that no model can eliminate. It's the variance of the irreducible error term.

A Curse of Dimensionality and Sample Complexity

Adaptive System In Upfal's book, he highlights that the fundamental concept of sampling complexity is to be able to sample from the smart space. It is crucial that we can sample from almost uniformly a solution to a "self-reducible" combinatorial problem, then we can construct a randomized algorithm that approximately estimates a solution. [30].

A.1 Variance Reduction

If we denote the exact mean by $\mu = \mathbb{E}[f(X)]$, and the MC approximation by $\hat{\mu}$, one can show that, with independent samples,

$$(\hat{\mu} - \mu) \to \mathcal{N}\left(0, \frac{\sigma^2}{n}\right)$$

where

$$\sigma^2 = \text{var}[f(X)] = \mathbb{E}\left[f(X)^2\right] - \mathbb{E}[f(X)]^2$$

This is a consequence of the central-limit theorem. Of course, σ^2 is unknown in the above expression, but it can also be estimated by MC:

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{s=1}^{S}\left(f\left(x_s\right) - \hat{\mu}\right)^2$$

Then we have

$$P\left\{\mu - 1.96\frac{\hat{\sigma}}{\sqrt{n}} \leq \hat{\mu} \leq \mu + 1.96\frac{\hat{\sigma}}{\sqrt{n}}\right\} \approx 0.95$$

The term $\sqrt{\frac{\hat{\sigma}^2}{n}}$ is called the (numerical or empirical) standard error, and is an estimate of our uncertainty about our estimate of μ.

If we want to report an answer which is accurate to within $\pm\epsilon$ with probability at least 95%, we need to use a number of samples S which satisfies $1.96\sqrt{\hat{\sigma}^2/n} \leq \epsilon$. We can approximate the 1.96 factor by 2 , yielding $n \geq \frac{4\hat{\sigma}^2}{\epsilon^2}$ [31][54-55].

.2 proof of error

$$\begin{aligned}
\mathbb{E}\left[(Y - \hat{Y})^2\right] &= \mathbb{E}\left[(f(X) + \varepsilon - \hat{f}(X))^2\right] \\
&= \mathbb{E}\left[f(X)^2 + \varepsilon^2 + \hat{f}(X)^2 + 2\varepsilon f(X) - 2\varepsilon\hat{f}(X) - 2f(X)\hat{f}(X)\right] \\
&= \mathbb{E}\left[f(X)^2 + \hat{f}(X)^2 - 2f(X)\hat{f}(X)\right] + \mathbb{E}\left[\varepsilon^2\right] + \underbrace{2\mathbb{E}[\varepsilon]\mathbb{E}[f(X) - \hat{f}(X)]}_{\text{independence}} \\
&= \mathbb{E}\left[(f(X) - \hat{f}(X))^2\right] + \underbrace{\mathbb{E}\left[\varepsilon^2\right] + \underbrace{\mathbb{E}[\varepsilon]^2}_{=0}}_{\text{variance}} \\
&= \underbrace{\mathbb{E}\left[(f(X) - \hat{f}(X))^2\right]}_{\text{Reducible error}} + \underbrace{\text{Var}(\varepsilon)}_{\text{Irreducible error}}
\end{aligned}$$

.3 Sample Complexity

A PAC Learning Framework

Definition 1 (PAC)**.** *A concept class $\mathcal{C}$ over input set X is PAC learnable*[1] *if there is an algorithm L, with access to a function* $\mathrm{Oracle}(C, \mathcal{D})$*, that satisfies the following properties: for every correct concept $C \in \mathcal{C}$, every distribution $\mathcal{D}$ on X, and every $0 < \epsilon, \delta \leq 1/2$, the number of calls that the algorithm L makes to the function* $\mathrm{Oracle}(C, \mathcal{D})$ *is polynomial in ϵ^{-1} and δ^{-1} , and with probability at least $1 - \delta$ the algorithm L outputs a hypothesis h such that*

$$\mathrm{P}_{\mathcal{D}}(h(x) \neq c(x)) \leq \epsilon$$

.

We first prove that any finite concept class is PAC learnable.

Theorem 1. *Any finite concept class $\mathcal{C}$ can be PAC learned with $n = \underline{1}\left(\ln|\mathcal{C}| + \ln\frac{1}{8}\right)$ samples.*

Definition 2. *A randomized algorithm gives an (ε, δ)-approximation for the value V if the output X of the algorithm satisfies*

$$\mathrm{P}(|X - V| \leq \varepsilon V) \geq 1 - \delta.$$

Theorem 2. *Let $X_1, \ldots, X_n$ be independent and identically distributed indicator random variables, with $\mu = \mathbf{E}[X_i]$. If $n \geq (3\ln(2/\delta))/\varepsilon^2\mu$, then*

$$\mathrm{P}\left(\left|\frac{1}{n}\sum_{i=1}^{m} X_i - \mu\right| \geq \varepsilon\mu\right) \leq \delta.$$

That is, m samples provide an (ε, δ)-approximation for μ

.

Definition 3. *Let w be the (random) output of a sampling algorithm for a finite sample space Ω. The sampling algorithm generates an ε-uniform sample of Ω if, for any subset S of Ω,*

$$\left|\mathrm{P}(w \in S) - \frac{|S|}{|\Omega|}\right| \leq \varepsilon.$$

A sampling algorithm is a fully polynomial almost uniform sampler (FPAUS) for a problem if, given an input x and a parameter $\varepsilon > 0$, it generates an ε-uniform sample of $\Omega(x)$ and runs in time that is polynomial in $\ln\varepsilon^{-1}$ and the size of the input x.

Proof. Let $c^* \in \mathcal{C}$ be the correct classification. A hypothesis h is said to be "bad" if $\Pr_{\mathcal{D}}(h(x) \neq c^*(x)) \geq \epsilon$. The probability that any particular bad hypothesis is consistent with m random samples is bounded above by $(1 - \epsilon)^m$, and hence the probability that any bad hypothesis is consistent with m random samples is bounded above by

$$|\mathcal{C}|(1 - \epsilon)^m \leq \delta.$$

□