Sensitivity Analysis and Uncertainty Estimation of FINN

TODO

December 2, 2024

Abstract

This study investigates uncertainty quantification (UQ) for retardation factor estimation in contaminant transport modeling using the Finite Volume Neural Network (FINN). Accurate retardation factor estimation is crucial for predicting contaminant fate and transport, but traditional methods rely on simplified isotherms that are inaccurate. FINN offers a data-driven alternative by learning complex retardation behavior from concentration data. However, existing UQ methods for FINN, such as Bayesian approaches, can be computationally demanding. We propose two novel, more efficient UQ methods: Stochastic Perturbation Analysis of Networks (SPAN) and Data-SPAN. SPAN explores uncertainty by training FINN with randomly sampled hyperparameters, while Data-SPAN utilizes randomly generated datasets using PI3NN. We empirically demonstrate the near-uniqueness of the retardation factor inverse problem, justifying our focus on solver and data uncertainties. Applying our methods to synthetic and experimental datasets, we construct prediction intervals for the retardation function and analyze the sensitivity of FINN to hyperparameter variations. We compare our methods with a Markov Chain Monte Carlo (MCMC) baseline, finding that while SPAN and Data-SPAN are computationally more expensive per sample, they offer comparable prediction interval quality and provide a more comprehensive exploration of the uncertainty landscape. Data-SPAN, in particular, shows promising results in terms of reliability. Our findings contribute to more efficient and robust UQ for FINN-based contaminant transport modeling.

Contents

1 Introduction

1.1 Motivation

Accurate prediction of contaminant transport in groundwater relies heavily on understanding sorption, often represented by retardation factors in transport models. Traditional models use predefined sorption isotherms, introducing uncertainty due to their limited ability to capture complex real-world behavior [5]. The finite volume neural network (FINN) [5] offers a data-driven alternative by using artificial neural networks (ANNs) to learn retardation factors as a function of concentration, combining the strengths of the finite volume method (FVM) and ANNs. This hybrid approach allows for capturing complex sorption phenomena while maintaining numerical stability and interpretability [5].

However, uncertainty quantification (UQ) for FINN, and machine learning models in general, remains a challenge. While existing methods like Bayes-by-backprop and Markov chain Monte Carlo (MCMC) are effective [5], they can be computationally expensive, hindering their practical application. This study addresses the need for more efficient UQ in FINN-based retardation factor estimation. Specifically, we propose a novel approach that assumes random hyperparameters of the ANN, rather than random weights, to significantly accelerate UQ while maintaining accuracy comparable to existing Bayesian methods. Our central hypothesis is that by focusing on hyperparameter uncertainty, we can achieve efficient and accurate UQ for FINN, making ML-based contaminant transport modeling more practical for real-world applications.

1.2 Background

The transport of contaminants in groundwater is a complex process that involves various physical and chemical interactions. One crucial aspect is diffusion-sorption, where contaminants dissolved in groundwater can interact with the solid phase of the porous medium, such as clay, leading to sorption and desorption processes. This interaction significantly affects the migration and fate of contaminants. The retardation factor (R) is a key parameter in diffusion-sorption models, quantifying the delay in contaminant transport caused by sorption. It represents the ratio of the contaminant's velocity in the absence of sorption to its actual velocity in the presence of sorption.

Mathematically, the diffusion-sorption process is often described by a partial differential equation (PDE) that incorporates the retardation factor. For example, Equation (1) presents a one-dimensional diffusion-sorption equation for dissolved contaminant concentration (c), where the retardation factor (R) is a function of c:

$$\frac{\partial c}{\partial t} = \frac{D}{R(c)} \frac{\partial^2 c}{\partial x^2} \tag{1}$$

Here, D represents the effective diffusion coefficient, t is time, and x is the distance along the flow path.

Determining the retardation factor from concentration data is an inverse problem. Instead of directly solving the PDE with known parameters, the goal is to infer the unknown retardation factor function ($R(c)$) from observed concentration data. This inverse problem is challenging due to the limited availability and potential noise in the data, as well as the complexity of the underlying sorption processes. Even without noise, uniqueness is not guaranteed for implicit equations. Therefore, we empirically investigate the uniqueness of the retardation factor function given the diffusion-sorption PDE and boundary conditions in section 2.

Uncertainty quantification (UQ) for neural networks is essential for providing reliable predictions and assessing the confidence in learned models. Given that neural networks learn from data, uncertainties in the data, model parameters, and model structure can propagate to the predictions. Traditional UQ methods for neural networks, such as Bayes-by-backprop, involve treating the network weights as random variables and estimating their posterior distribution. However,

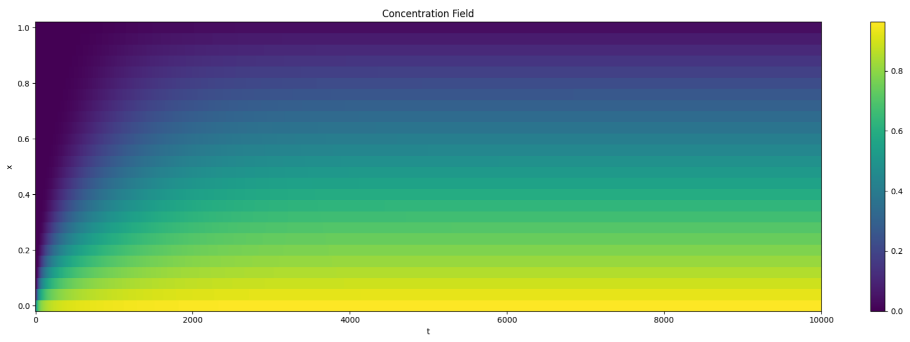

Figure 1: Synthethic concentration field.

these methods can be computationally expensive, especially for complex models. Our approach of considering random hyperparameters offers a potentially more efficient alternative for estimating uncertainty in the learned retardation factor.

1.3 Problem Statement

The diffusion-sorption process, governed by a PDE, describes contaminant transport in groundwater. Experimentally, data can be obtained from a soil cylinder, including breakthrough curves (BTCs) at a fixed location ($x = L$) over time ($t \in [0, T_{\text{end}}]$), and a concentration profile at a specific time ($t = T_{\text{end}}$) across the cylinder's length ($x \in [0, L]$). While a complete concentration field ($c(x, t)$, see Figure 1) would be ideal, it's practically unobtainable as measurements at all x values require destructive sampling. Solving the PDE requires knowing the retardation function, $R(c)$, which quantifies sorption and is dependent on soil properties. Since $R(c)$ is typically unknown, FINN is employed to learn it from the available data. However, uncertainties arise from measurement errors in the data and non-uniqueness of the solver solution, leading to uncertainty in the learned retardation function. Therefore, the goal is to estimate a prediction interval (PI) around the learned $R(c)$ to quantify this uncertainty.

1.3.1 Learning

The goal of FINN is to learn the unknown retardation factor, $R(c)$, which is a function of the contaminant concentration, c. The retardation factor is a part of the diffusion-sorption equation (1) with boundary conditions:

- Top Boundary: At the top end of the sample where pure-phase TCE is injected, a Dirichlet boundary condition is used:

$$c|_{x=0} = c_{sol} \quad \forall t : 0 \leq t \leq T, \tag{2}$$

  where c_{sol} is the solubility limit of TCE in water and T is the experiment time.

- Bottom Boundary: At the bottom end of the sample, which is flushed with clean water, a Cauchy boundary condition is applied:

$$c|_{x=L} = \frac{D}{Q}\frac{\partial c}{\partial x} \quad \forall t : 0 \leq t \leq T, \tag{3}$$

where Q is the flow rate of the clean water and L is the sample length.

The goal is to learn $\hat{R}(x, t; \theta)$ (represented by a neural network) given the data $\mathcal{D} = \{(x_i, t_i, c_i)\}_{i=1}^{N}$ which consists of concentration measurements c_i at spatial positions x_i and temporal points t_i. The network needs to be trained such that the PDE solution using $\hat{R}$ minimizes the error with respect to the data.

1.3.2 Uncertainty Quantification

There are two main aspects of uncertainty quantification (UQ) in this context:

1. UQ for FINN output: $p(\hat{c}(x, t)|\mathcal{D})$. This represents the probability distribution of the predicted contaminant concentration $\hat{c}(x, t)$ obtained from the FINN model, given the data $\mathcal{D}$.
2. UQ for the predicted retardation factor: $p(\hat{R}(c)|\mathcal{D})$. This is arguably more important because it represents the probability distribution of the predicted retardation factor $\hat{R}(c)$ given the data $\mathcal{D}$. This distribution helps in understanding the confidence in the estimated retardation factor.

But there are also two types of uncertainty that have to be differentiated [1, 2]:

1. Aleatoric uncertainty: This arises from inherent randomness in the data and the physical system. It is irreducible and is caused by factors such as measurement noise and variability in the system parameters.
2. Epistemic uncertainty: This stems from limited knowledge of the model (e.g. uncertainty over its parameters). It is associated with model structure, neural network architecture, and limited training data. It quantifies the model's predictive uncertainty.

While a clear decomposition into aleatoric and epistemic uncertainty is often not feasible, our interest extends to quantifying the overall predictive uncertainty, which encompasses both.

1.4 Contributions of this Work

This study addresses uncertainty quantification (UQ) in the finite volume neural network (FINN) framework when applied to a diffusion-sorption process. We begin with an empirical analysis of the uniqueness of the retardation factor, $R(c)$, which is crucial for interpreting the FINN output and forms the basis of our proposed UQ approach. We summarize FINN and PI3NN, the foundational methods used in this work. A novel framework is introduced to assess both solver and data uncertainty, allowing us to construct prediction intervals (PIs) for the retardation function. Additionally, our method performs a sensitivity analysis, exploring perturbation-sensitive directions and their impact on the FINN output. We apply the proposed framework to both synthetic and experimental datasets, comparing the results with established baselines such as the Markov Chain Monte Carlo (MCMC) approach. Our method involves using random hyperparameters of the neural network for UQ as well as random datasets. This efficient approach aims to capture both aleatoric and epistemic uncertainties, offering a computationally less expensive alternative to traditional Bayesian methods by focusing on hyperparameter variations instead of the high-dimensional weight space.

1.5 Related Work

1.5.1 Bayes NN

This section describes the baseline methodology used for comparison, which employs Bayesian inference to estimate the posterior distribution of the retardation function, $\hat{R}(c(x,t))$, parameterized by a neural network (NN). The retardation function is a function of concentration, c, which itself is a function of spatial position, x, and time, t. This approach updates prior beliefs about the retardation function based on observed data, $\mathcal{D}$.

Prior Distribution: A Gaussian prior distribution, $p(\theta)$, is placed over the NN parameters θ, reflecting prior knowledge or assumptions. This prior is centered on the parameters θ_{PT} of a pre-trained NN, with covariance matrix $\Sigma_p = 0.05\,I$:

$$p(\theta) = \mathcal{N}(\theta|\theta_{PT}, \Sigma_p)$$

Likelihood Function: The likelihood function, $p(\mathcal{D}|\theta)$, quantifies the probability of observing the data $\mathcal{D}$ given a specific set of NN parameters θ. We assume additive Gaussian noise with standard deviation σ corrupts the observed concentrations. Let $c_{obs}(x,t)$ represent the observed concentration at position x and time t, and $c_{model}(x,t;\theta)$ the concentration predicted by the model (using a FINN solver with $\hat{R}(c(x,t);\theta)$). The likelihood is:

$$p(\mathcal{D}|\theta) = \prod_{(x_i,t_i,c_i)\in\mathcal{D}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(c_i - c_{model}(x_i,t_i;\theta))^2}{2\sigma^2}\right) \tag{4}$$

Posterior Distribution: Bayes' theorem provides the posterior distribution of the NN parameters given the data:

$$p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$$

Praditia et al. [5] compare several methods to obtain samples from this distribution, including the Metropolis-Hastings algorithm, the MALA algorithm, and the Barker method. The accepted samples $\{\theta_i\}$ from the used algorithm are then treated as draws from the posterior distribution $p(\theta|\mathcal{D})$.

Posterior of Retardation Function: Finally, the posterior distribution of the retardation function $\hat{R}(c)$ is obtained by evaluating the NN with the sampled parameters $\{\theta_i\}$:

$$\{\hat{R}(c|\theta_i)\} \sim p(\hat{R}(c)|\mathcal{D})$$

This set of retardation function samples provides a probabilistic description of the retardation behavior, incorporating the uncertainty arising from the limited and noisy data.

Since Praditia et al. [5] obtain the best results with the Barker method and no burn-in period by starting from the pre-trained model, we follow their approach to have a fair comparison.

1.6 Outline of this Work

This work is structured as follows: Section 2 empirically investigates the uniqueness of the retardation factor inverse problem. Section 3 provides background information on the employed methods FINN and PI3NN. Section 4 details our methodology, introducing the Stochastic Perturbation Analysis of Networks (SPAN) and Data-SPAN approaches for uncertainty quantification. Section 5 describes the experimental data and setup. Section 6 presents the experimental results, including runtime comparisons and evaluations of likelihood and reliability. Section 7 discusses the findings and limitations of the proposed methods. Finally, Section 8 summarizes the work and outlines potential future research directions.

2 Uniqueness of Inverse Problem

Inverse problems are not necessarily unique. Consider the simple equation $x^2 = 4$. This equation has two solutions, $x = 2$ and $x = -2$. Thus, knowing the output (4) does not uniquely determine the input (x). This illustrates the concept of non-uniqueness in inverse problems. Instead of rigorously analyzing the problem mathematically, we perform an empirical uniqueness analysis. We achieve this by rearranging the PDE (1) to solve for $R(c)$

$$R(c) = D\frac{\partial^2 c}{\partial x^2}\frac{1}{\frac{\partial c}{\partial t}}$$

and substituting the synthetic data. The results, shown in Figure 2, suggest a unique solution for this data, except for minor numerical inaccuracies.

This allows us to attribute potential discrepancies in solutions to two primary factors: solver uncertainty and data uncertainty. We will first examine these separately and then combine them to obtain a final uncertainty quantification.

3 Foundational Methods

3.1 FINN

The Finite Volume Neural Network (FINN) is a specialized machine learning approach that combines the strengths of traditional numerical methods with the adaptability of neural networks to solve partial differential equations (PDEs) with unknown or uncertain components. It leverages the Finite Volume Method (FVM), dividing the problem domain into discrete control volumes. The core of FINN lies in its use of ”flux kernels” F_i – neural networks that learn how quantities flow between neighboring volumes. Specifically, each volume has a flux kernel composed of two subkernels, f_{i-} and f_{i+}, one for each direction, which model these fluxes based on the concentrations in the volume and its neighbors. These subkernels consist of two parts: a ”stencil” component Φ_N that approximates the FVM spatial scheme and a ”diffusivity” component Φ_D that learns how the flow rate depends on the concentration itself:

$$\begin{aligned} f_{i-} &= \Phi_D(c_i) \cdot \Phi_N(c_i, c_{i-1}) \\ f_{i+} &= \Phi_D(c_i) \cdot \Phi_N(c_i, c_{i+1}). \end{aligned}$$

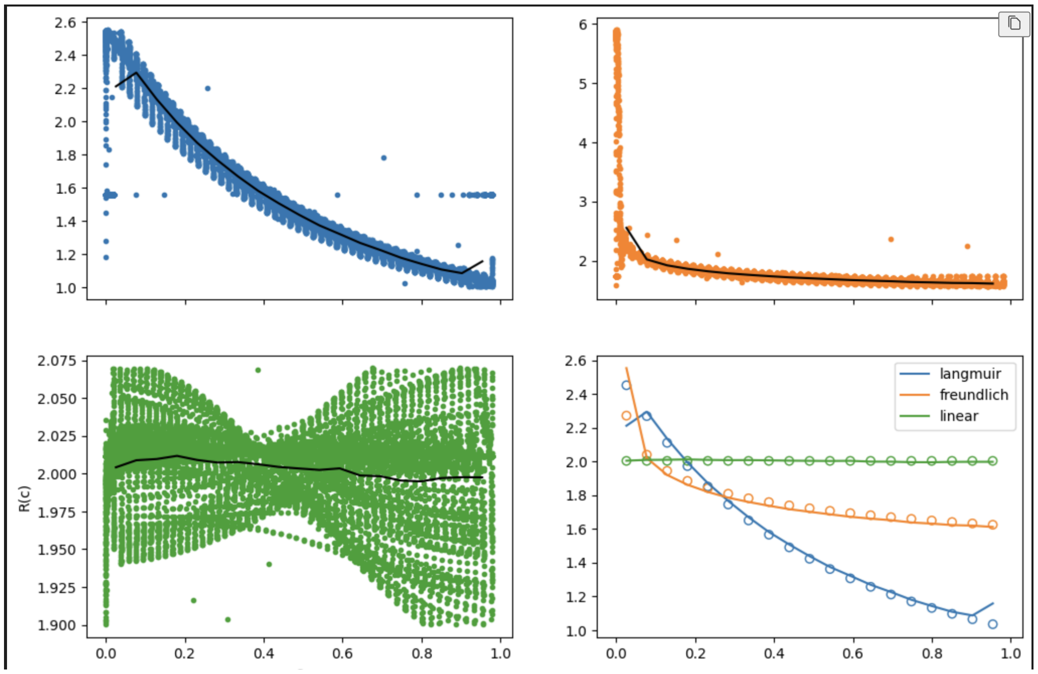

Figure 2: Computed retardation values (dots) and mean (black line) from analytical expression and synthetic data for linear, Freundlich and Langmuir retardations. Bottom right: Means (colored lines) and exact synthetic retardations (circles).

Furthermore, FINN employs "state kernels" S_i that take the local concentration and calculated fluxes as input to update the concentration over time. This time evolution is governed by a Neural Ordinary Differential Equation (NODE) solver, a differentiable method that integrates seamlessly into the neural network training. By integrating FVM discretization, specialized neural network modules, and a NODE solver, FINN is capable of learning complex time-dependent spatial patterns while adhering to physical conservation laws and boundary conditions.

3.2 PI3NN

PI3NN (Prediction Intervals from Three Neural Networks) [3] is a method for constructing prediction intervals (PIs) that uses three independently trained neural networks. It aims to provide tight, non-crossing PIs across various confidence levels without requiring retraining for each level.

The method's theoretical foundation lies in approximating the ground-truth PI bounds with a family of neural networks. Specifically, PI3NN approximates the median of the target variable, $M[y]$, and the expected deviations above and below the median, $E[(y-M[y])\nVdash_{\{y-M[y]>0\}}]$ and $E[(M[y]-y)\nVdash_{\{M[y]-y>0\}}]$, respectively, using three independently trained neural networks ($f_w(x)$, $u_\theta(x)$, and $l_\epsilon(x)$). These networks are trained using the standard mean squared error loss, simplifying the training process and avoiding the need for specialized loss functions and sensitive hyperparameters that often require fine-tuning in other PI methods.

Once trained, PI3NN constructs the PI bounds as linear combinations of the three networks' outputs. The coefficients of these linear combinations (α and β) are determined through a root-finding algorithm that ensures the desired PICP (Prediction Interval Coverage Probability) for a given confidence level q (γ in [3]). This process allows for calculating PIs for multiple confidence levels without retraining the networks, offering significant computational advantages.

Figure 3 illustrates the 6-step breakdown of the PI3NN process:

1. **Learn the mean:** Train a neural network to approximate the mean of the target data.
2. **Estimate the median:** Calculate the median by shifting the learned mean.
3. **Calculate and split residuals:** Compute the residuals between the actual data and the estimated median, then separate these into positive and negative residuals.
4. **Learn residuals:** Train two more neural networks, one to approximate the positive residuals and the other to approximate the negative residuals.
5. **Construct PI:** Calculate a PI using the learned median and the learned positive and negative residuals.
6. **Generate multiple PIs:** Use a root-finding algorithm to compute PIs for different confidence levels q without retraining the networks.

4 Methodology

4.1 SPAN (Stochastic Perturbation Analysis of Networks)

Our objective is to determine the posterior predictive distribution $p(\hat{R}(c;\theta) = R(c)|\mathcal{D})$, which represents the probability of computing a specific retardation function value $R(c)$ for a given concentration c, based on the observed breakthrough curve data $\mathcal{D}$. We present a method that computes

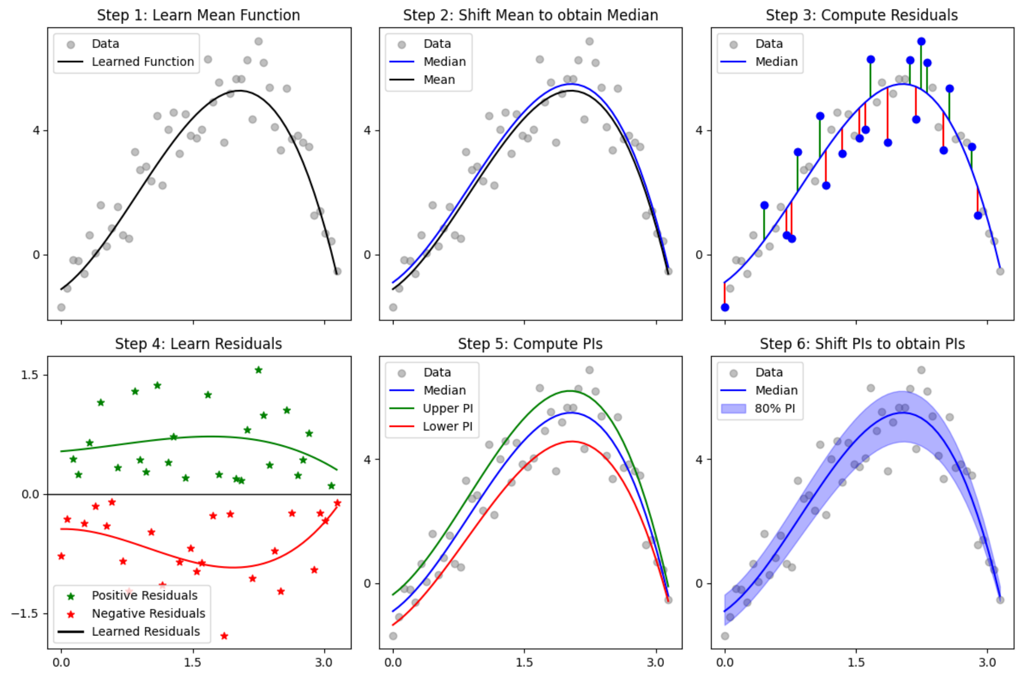

Figure 3: Illustration of the PI3NN method.

this distribution by marginalizing over uncertain parameters h of the solver and employing an approximate likelihood function $p(\mathcal{D}|h)$ based on the same assumtions that were used for the Bayes aproach.

4.1.1 Approximating the Posterior Predictive Distribution via Hyperparameter Marginalization

Model Formulation: We represent our neural network as $\hat{R}(c;\theta)$, where c is the input concentration and θ denotes the network's parameters (weights and biases). The parameters θ are determined by a deterministic solver $S(h,\mathcal{D})$, which takes hyperparameters h and training data $\mathcal{D}$ as input. A prior distribution $p(h)$ is assigned over the hyperparameters; the exact form of $p(h)$ is not crucial, as it suffices to generate samples from it.

With $w(h) = \frac{p(\mathcal{D}|h)}{p(\mathcal{D})}$, the posterior predictive distribution can be expressed as:

$$\begin{aligned} p(\hat{R}(c;\theta) = R(c)|\mathcal{D}) &= \int p(\hat{R}(c;\theta)|h,\mathcal{D})\; p(h|\mathcal{D})\, dh \\ &= \int p(\hat{R}(c;\theta)|h,\mathcal{D})\; \frac{p(\mathcal{D}|h)p(h)}{p(\mathcal{D})}\, dh \\ &= \int p(\hat{R}(c;\theta)|h,\mathcal{D})\; w(h)\; p(h)\, dh \\ &= \int \delta(\hat{R}(c;\theta) - R(c))\; w(h)\; p(h)\, dh \end{aligned}$$

The first step is the marginalization over h, the second the application of Bayes' Theorem, and the last stems from the fact that the solver $\theta = S(h,\mathcal{D})$ is deterministic given h and $\mathcal{D}$ and thus the distribution becomes a delta distribution.

Importance Sampling: So sampling from $p(\hat{R}(c;\theta) = R|\mathcal{D})$ becomes equivalent to sampling from $p(h)$, evaluating the solver to obtain the model parameters $\theta = S(h,\mathcal{D})$, evaluating the model $\hat{R}(c;\theta)$, and reweighting the samples with $w(h)$. This is the concept of importance sampling. To approximate the distribution, we thus have to draw M samples $\{h_m\}_{m=1}^{M}$ from the prior distribution $p(h)$ and calculate the importance weight for each:

$$w_m = \frac{p(\mathcal{D}|h_m)}{p(\mathcal{D})} \propto p(\mathcal{D}|h_m)$$

As $p(\mathcal{D})$ is a constant, it can be omitted, and the weights can be normalized subsequently.

Likelihood Computation: We use the same assumption about our data that the Bayesian baseline uses which results in an easy computation of the likelihood by just substituting θ with $S(h,\mathcal{D})$ in equation (4):

$$p(\mathcal{D}|h) = \prod_{x_i,t_i,c_i\in\mathcal{D}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(c_i - c_{model}(x_i,t_i;S(h,\mathcal{D})))^2}{2\sigma^2}\right)$$

4.1.2 Data-SPAN

The exact same derivation can be applied when we assume a random dataset $\tilde{\mathcal{D}}$ instead of random hyperparameters h:

$$\begin{aligned} p(\hat{R}(c;\theta) = R(c)|\mathcal{D}) &= \int p(\hat{R}(c;\theta)|\tilde{\mathcal{D}},\mathcal{D})\ p(\tilde{\mathcal{D}}|\mathcal{D})\,d\tilde{\mathcal{D}} \\ &= \int p(\hat{R}(c;\theta)|\tilde{\mathcal{D}},\mathcal{D})\ \frac{p(\mathcal{D}|\tilde{\mathcal{D}})p(\tilde{\mathcal{D}})}{p(\mathcal{D})}\,d\tilde{\mathcal{D}} \\ &= \int p(\hat{R}(c;\theta)|\tilde{\mathcal{D}},\mathcal{D})\ w(\tilde{\mathcal{D}})\ p(\tilde{\mathcal{D}})\,d\tilde{\mathcal{D}} \\ &= \int \delta(\hat{R}(c;\theta) - R(c))\ w(\tilde{\mathcal{D}})\ p(\tilde{\mathcal{D}})\,d\tilde{\mathcal{D}} \end{aligned}$$

$$p(\mathcal{D}|\tilde{\mathcal{D}}) = \prod_{x_i,t_i,c_i\in\mathcal{D}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(c_i - c_{model}(x_i,t_i;S(h,\tilde{\mathcal{D}})))^2}{2\sigma^2}\right)$$

Random Dataset Sampling: To generate random datasets, we leverage PI3NN, applying it to the original breakthrough curve (BTC) dataset $\mathcal{D}$. Let $\hat{c}(t;\theta_{\mathcal{D}})$ be the mean concentration curve predicted by FINN trained on the full dataset $\mathcal{D}$, with $\theta_{\mathcal{D}}$ representing the learned FINN parameters. This FINN-predicted mean curve is used as input to the PI3NN algorithm, as it provides a more accurate representation of the underlying process compared to the mean curve learned by a standard MLP within the PI3NN framework.

PI3NN then learns the residuals, effectively modeling the distribution of c given t. Let $F^{-1}(q|t)$ denote the quantile function predicted by PI3NN, where $q \in [0,1]$ is the quantile level and t is the time point. This function provides the concentration value at time t corresponding to the q-th quantile of the learned distribution, see 4.

To create a random dataset sample $\tilde{\mathcal{D}}(q)$, we sample a quantile level q from a uniform distribution $\mathcal{U}(0,1)$. Then, the random dataset sample is constructed as:

$$\tilde{\mathcal{D}}(q) = \{(t_i, F^{-1}(q|t_i))\}_{i=1}^{N}$$

This process is repeated M times to obtain a collection of random datasets $\{\tilde{\mathcal{D}}_j\}_{j=1}^{M}$. Each $\tilde{\mathcal{D}}_j$ is then used to train a separate instance of FINN, effectively providing a set of retardation function samples, analogous to the samples obtained from the hyperparameter perturbation approach.

To improve the approximative likelihood computation in 7.3, we artificially insert the quantile levels 0 and 1 into the random samples. This improves the distribution fit on the histogram of samples.

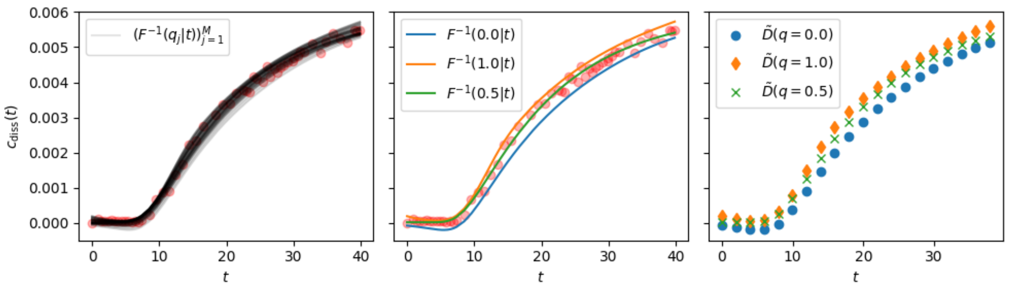

Figure 4: Left: BTC quantile functions for all quantile samples q_j. Middle: BTC quantile functions for quantiles 0, 0.5, and 1. Right: BTC datasets for quantiles 0, 0.5, and 1.

5 Data and Setup

5.1 Experimental Setup

The experiments were conducted using Python 3.11.3. Modified versions of the FINN code from Praditia et al. [5] and the PI3NN code from Liu et al. [3] were used. The code and specific library versions are available on GitHub. GNU parallel [6] was used for parallel execution of experiments.

5.2 Data Acquisition

The BTC data was obtained from the work by Praditia et al. [5], originally sourced from Nowak and Guthke [4]. Synthetic data was generated using the same solver employed by FINN, ensuring consistency in the experimental setup.

5.3 Training Details

Early stopping was implemented to halt training when the loss plateaued. Training runs for FINN were discarded if the normalized mean squared error (NMSE) exceeded 10^{-5} (measured on BTC), ensuring a minimum level of accuracy in the learned retardation function.

$$\text{NMSE}(y, \hat{y}) = \text{MSE}(y, \hat{y})/\text{mean}(y)$$

6 Experiments

6.1 SPAN

The hyperparameters that were used differ for the synthetic data case and the experimental data case:

In the experimental setting, 780 samples were generated using the specified hyperparameters. For the synthetic experiments, the number of samples varied and is indicated in the respective figure captions for clarity and comparison within those specific contexts. The experimental results are shown in 10.

Hyperparameter	Description	Experimental	Synthetic
Weight Initialization	Seed for initial values for model weights	✓	✓
Training Data Mask (see 5)	Seed for mask applied to training data	✗	✓
Training Data Noise	Seed for noise added to training data	✗	✓
Max Epochs	Maximum number of training iterations	✓	✓
MSE Loss Factor	Factor for Mean Squared Error loss	✓	✗
Physical Loss Factor	Factor for physical loss	✓	✗

Table 1: Hyperparameters for Experimental and Synthetic Settings

6.2 Data-SPAN

This approach was applied only to the experimental data because the synthetic training data lacks noise, leading to extremely small residuals and a very narrow distribution that does not impact sampling.

For the experimental data, we sampled 70 quantiles as detailed in 4.1.2. The results are shown in 11.

6.3 Full-SPAN

This approach is equivalent to Data-SPAN but additionally, for every sampled quantile, the hyper-parameters are also randomly sampled. The results are shown in 12.

6.4 Baselines (Bayes NN via MCMC)

We computed 10k samples using the same parameters as Praditia et al. [5]; the samples are visualized in 6.

7 Results

7.1 Samples and PIs

7.1.1 Synthetic Data

SPAN When the hyperparameters are randomly sampled to learn different retardations, the uncertainty in $R(c)$ is very small as can be seen in Figure 7, even when all hyperparameters are

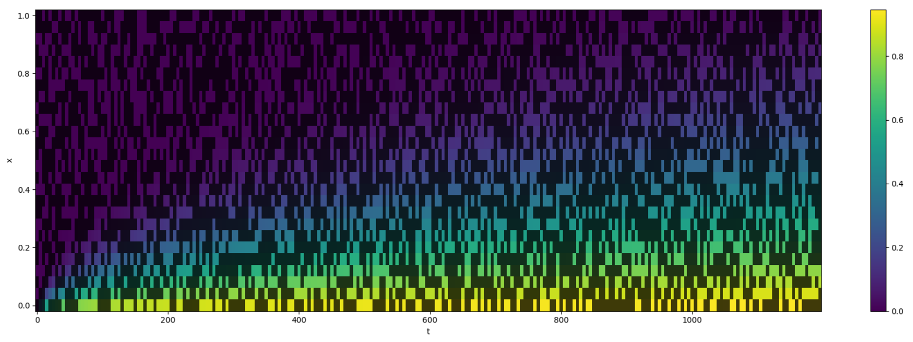

Figure 5: Full synthetic concentration training data where 50% of datapoints are randomly masked (dark patches).

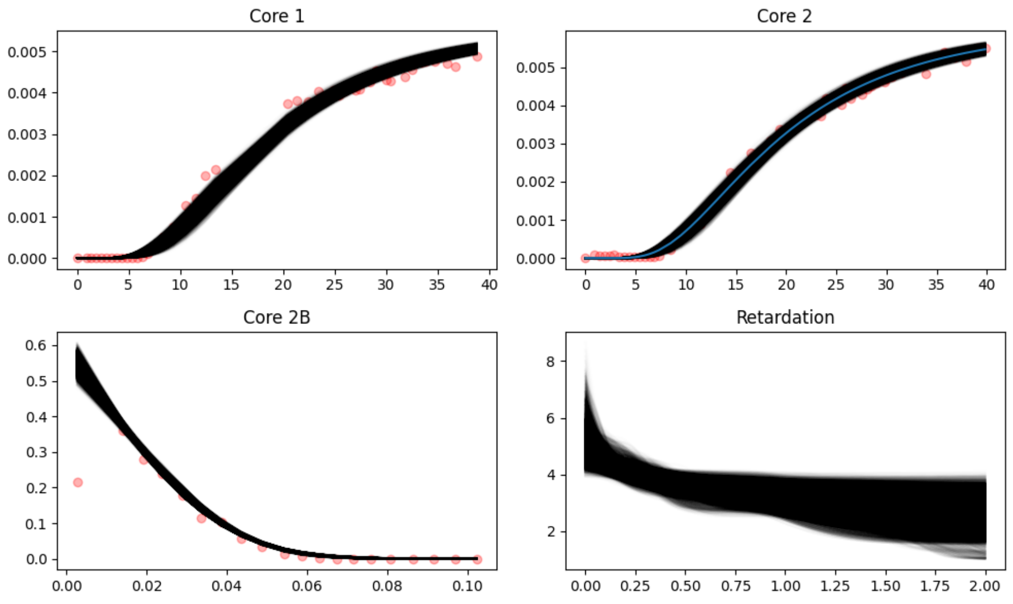

Figure 6: Retardation curves generated by MCMC sampling as detailed in 1.5.1.

combined in "All UQ".

Data-SPAN When applying gaussian noise of the same strength as the experimental data has due to measurement error [4], the uncertainty in $R(c)$ becomes much larger compared to SPAN as can be seen in Figure 8.

7.1.2 Experimental Data

Figure 9 illustrates that the 90% prediction intervals (PIs) for both the Markov Chain Monte Carlo (MCMC) and full SPAN methods encompass similar concentration values. This agreement arises from their comparable coverage of $R(c)$ values when $c \leq 1$. While the methods diverge for larger values of c, this discrepancy has minimal impact on the concentration fields, as these are less sensitive to $R(c)$ in that regime. Given that full SPAN produces similar c PIs but incorporates larger uncertainty in R, it provides a more robust uncertainty estimate. A more quantitative exploration of these findings is presented in the following sections.

7.2 Runtime

We measure and compare the runtime of generating a single sample of our method and the baseline. Since SPAN as well as Data-SPAN require training of the NN from scratch for each sample, the runtime approximately equals the training time. Averaging over 10 trials, the runtime is 170 seconds.

The MCMC method uses thinning, saving only every 10th sample. The average runtime for this is about 11.4 seconds, almost 15 times faster than our method.

7.3 Likelihood

To evaluate the predictive quality of our method, we use its samples to estimate a probability distribution. This is done by computing a histogram. We can then estimate the likelihood of the training data $\mathcal{D}$ given this distribution. We transform the likelihood into a negative log-likelihood (NLL) to obtain more accurate results. A lower NLL is better. Applying this procedure on SPAN yields a NLL of -6.41, -7.11 for MCMC, and -7.14 for Data-SPAN. A good baseline to compare this against, is a normal distribution around the FINN BTC prediction mean and standard deviation equal to the sample standard deviation computed from the residuals:

$$p(c;t) = \frac{1}{\sqrt{2\pi\int^2}} \exp(-\frac{1}{2\int^2}(\hat{c}(x = L, t; \theta_{\mathcal{D}}) - c)^2)$$

This yields a NLL of -7.50, which is the lowest value.

7.4 Reliability Curve

Another quantification done by Praditia et al. [5] is the average calibration of the PI. This can be visualized using the reliability curve.

The reliability curve assesses the calibration of predicted confidence intervals. It plots the observed frequency of the true value falling within a given confidence interval against the predicted confidence level. To compute it, the prediction range is divided into bins (e.g., by confidence levels).

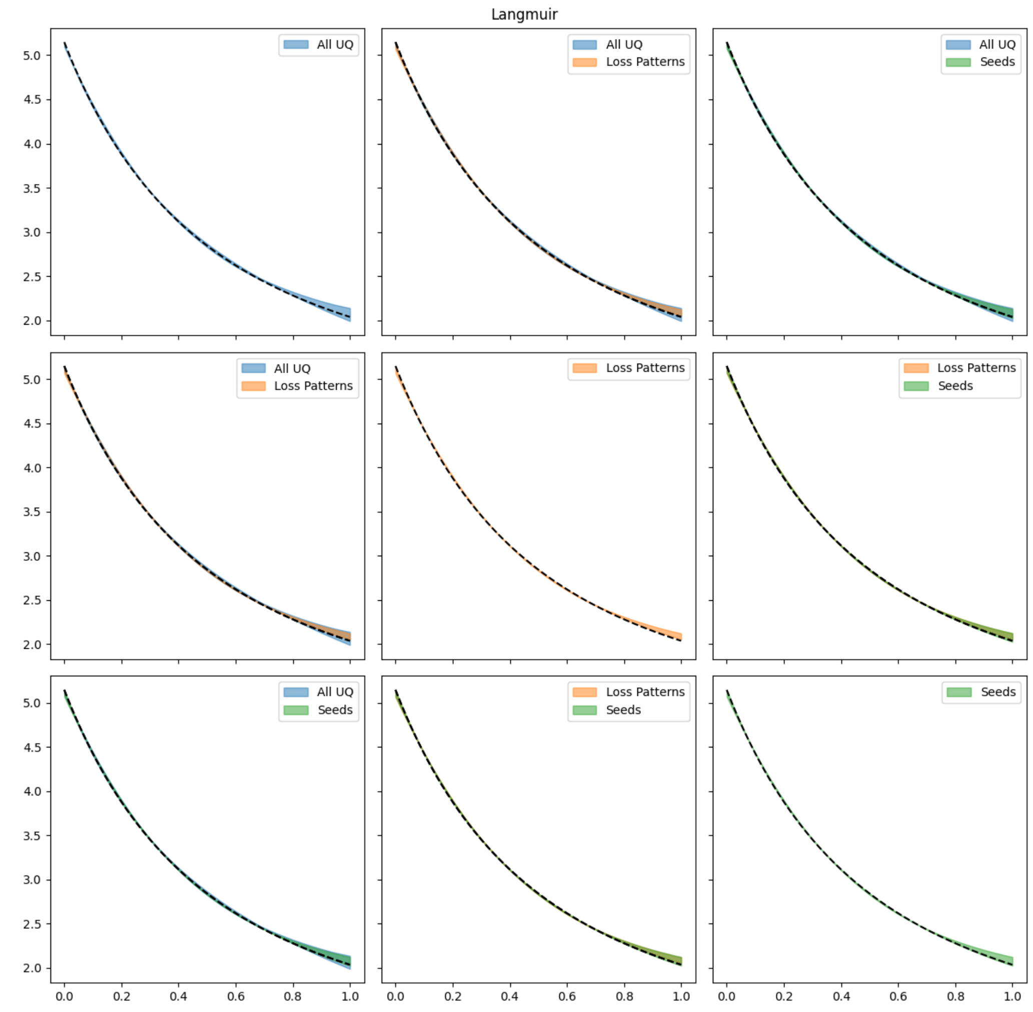

Figure 7: Comparison of retardation PIs obtained via SPAN on synthetic data (generated by langmuir isotherm).

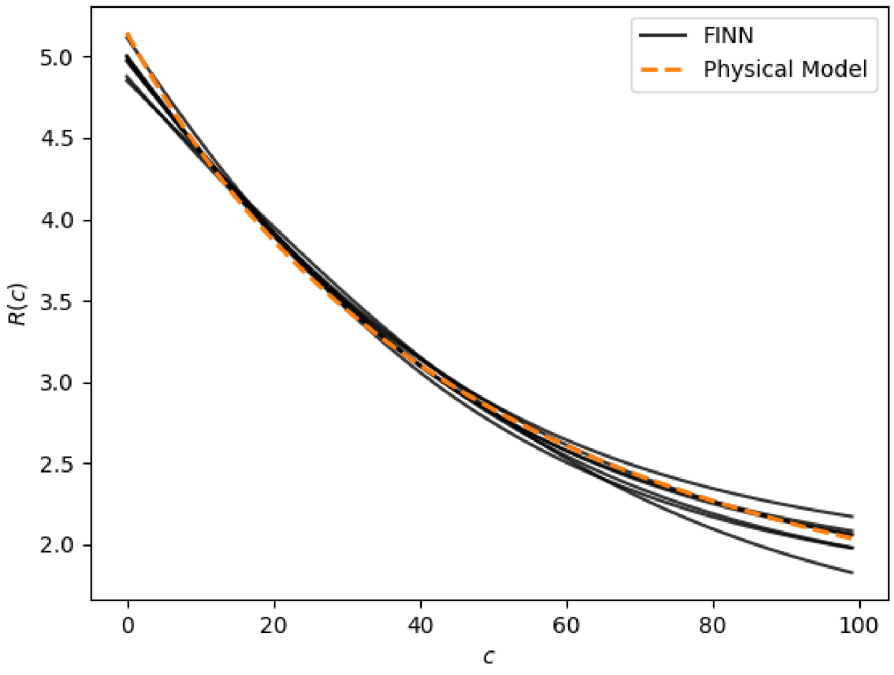

Figure 8: Retardations learned by FINN on synthetic dataset (generated by langmuir isotherm) pertubed by gaussian noise.

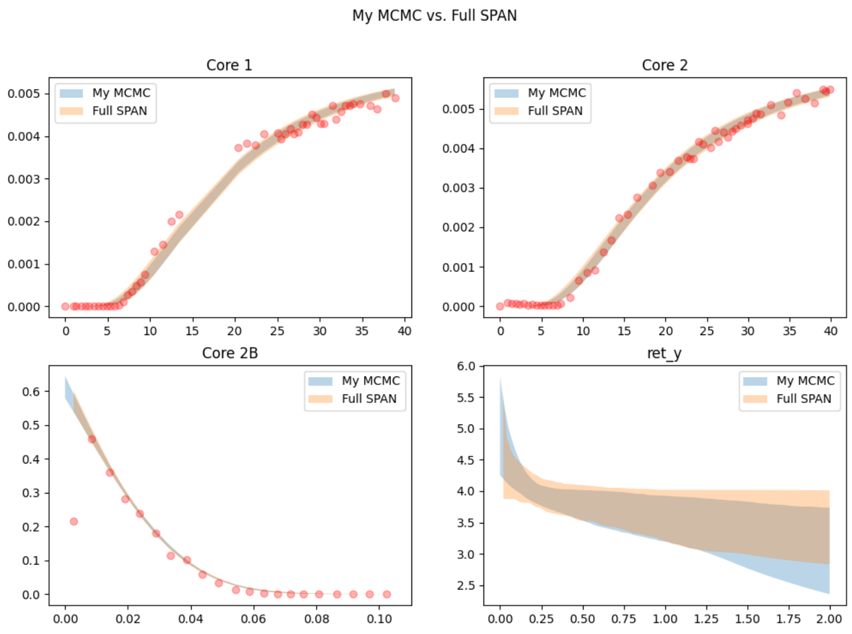

Figure 9: Retardation PIs and correspoding concentration PIs obtained from it by training FINN with random hyperparameters and random datasets using PI3NN (orange). Compared with MCMC approach (blue). (90% PIs are depicted.)

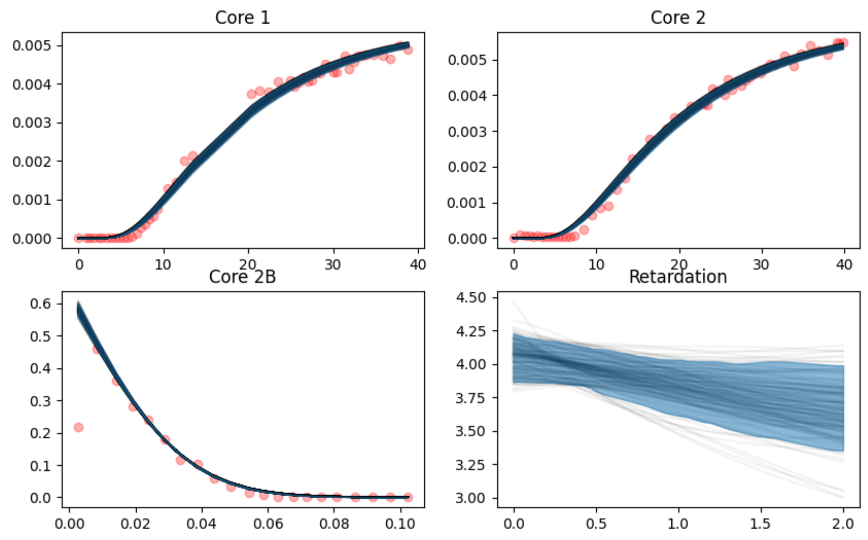

Figure 10: Retardation samples and correspoding concentration curves obtained from it by training FINN with random hyperparameters.

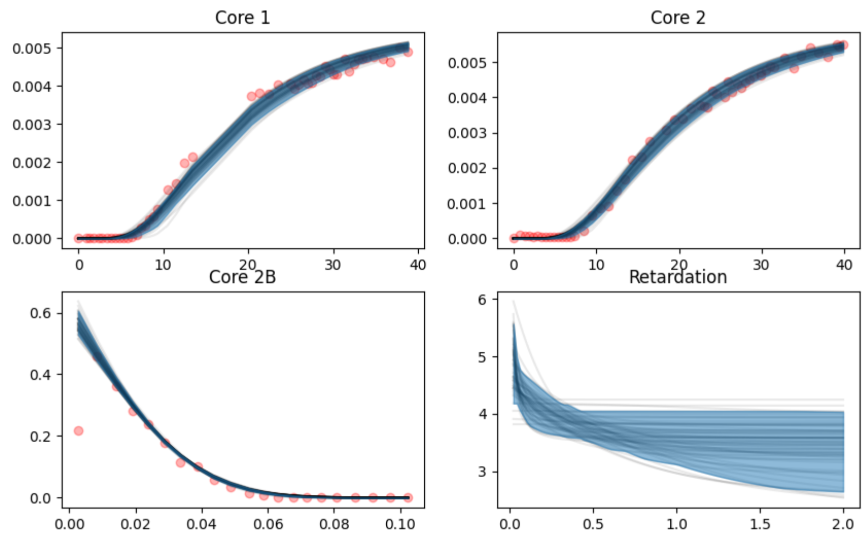

Figure 11: Retardation samples and correspoding concentration curves obtained from it by training FINN with random datasets using PI3NN.

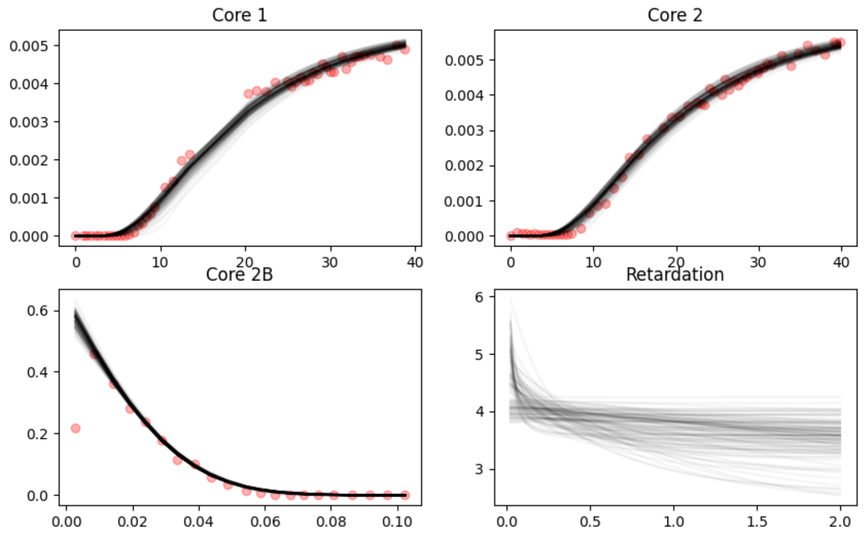

Figure 12: Retardation samples and correspoding concentration curves obtained from it by training FINN with random hyperparameters and random datasets using PI3NN.

For each bin, the proportion of predictions where the true value falls within the corresponding PI is calculated. Ideally, the curve should follow the diagonal line (perfect calibration). A curve above the diagonal indicates underconfidence (the true value falls within the PI more often than predicted), while a curve below the diagonal indicates overconfidence.

The results, shown in Figure 14, suggest that Data-SPAN provides the best calibrated PI. Overall, all methods are generally overconfident, SPAN the strongest. Only Data-SPAN is a little underconfident for very small c values.

7.5 Sensitivity Analysis

Sensitivity analysis investigates how variations in input parameters affect the model output. Unweighted SPAN samples inherently perform sensitivity analysis because each sample represents a different set of hyperparameters, and the resulting retardation function and its impact on the concentration field demonstrate the influence of those hyperparameters.

One sensitivity analysis method is based on standardized regression coefficients, sometimes referred to as beta coefficients or beta weights. They represent the relative importance of each predictor variable (hyperparameter h in this case) in explaining the variance of the outcome variable ($R(c)$). Importantly, the sum of these normalized coefficients can exceed 1 as parameters may explain common variances. The value shows how important the respective parameter is in relation to the rest. Since these parameters are based on linear regression, they should be interpreted with caution, especially when non-linear relationships exist between the hyperparameters and the retardation function.

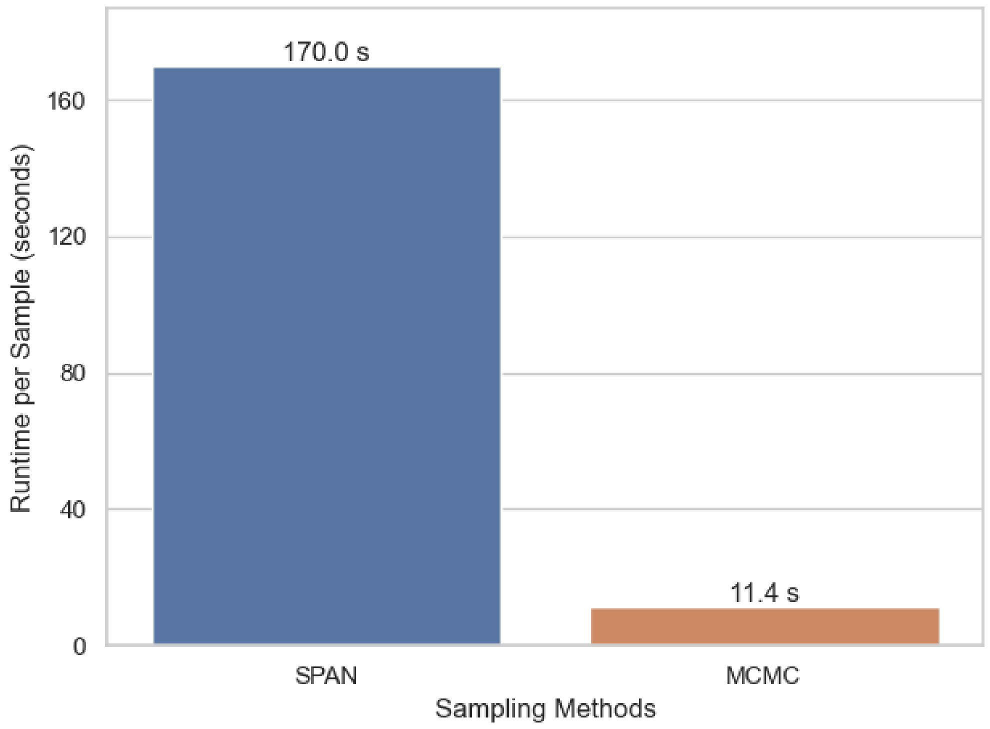

Figure 13: Runtime for generating a single sample using SPAN (left bar) and MCMC (right bar). Measured on the same machine and averaged over 10 trials. Variance is too insignificant to matter here.

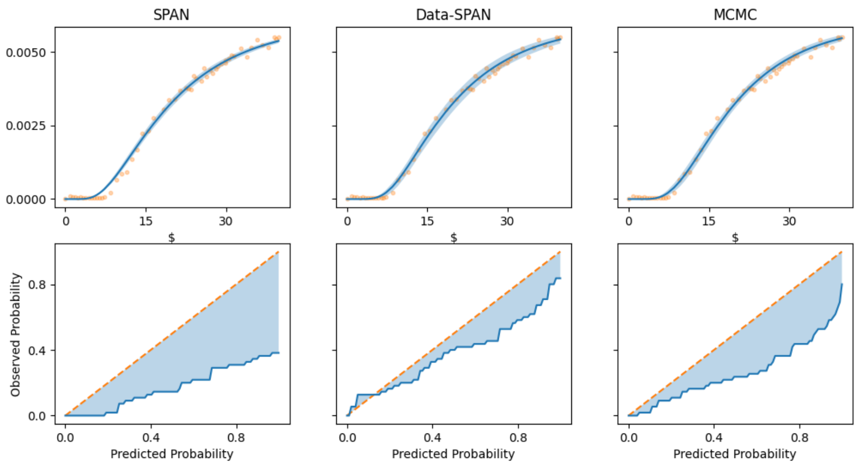

Figure 14: Top: Core 2 BTC 90% PIs for the different methods (SPAN, Data-SPAN, MCMC). Bottom: Reliability curve for each method.

We average the importance for $R(c)$ over a set of c values and obtain 0.033 for **Physical Loss Factor**, 0.051 for **Weight Initialization**, 0.159 for **MSE Loss Factor**, and 0.570 for **Max Epochs**.

This indicates that **Max Epochs** and **MSE Loss Factor** have the largest impact on $R(c)$ overall. The seed used for weight initialization and the physical loss factor play only negligible roles.

Figure 15 shows the relative importance for values of c over which the average was taken.

8 Discussion

8.1 Retardation Range

For some c $R(c)$ does not affect the whole c field equally. This can be seen by looking at figure 16 which shows constant retardation curves that are pertubed additively by triangle functions with centers at different c values. With increasing c center, the error caused by the hat functions goes quickly to zero. For $c > 1$, the error becomes zero. This is because the concentration field $c(x,t)$ only contains values between 0 and 1. But there is also a noticeable difference between the maximum error measured on the full field vs. the BTC. The BTC error is consistently much lower. The full field error can not be detected in the case of experimental data as there is no access to the full field solution. For this reason, the range of c values for which significant statements about the uncertainty of $R(c)$ can be made, is restricted to a narrow range. The exact range is difficult to estimate as the stength of influence for different retardations is unclear.

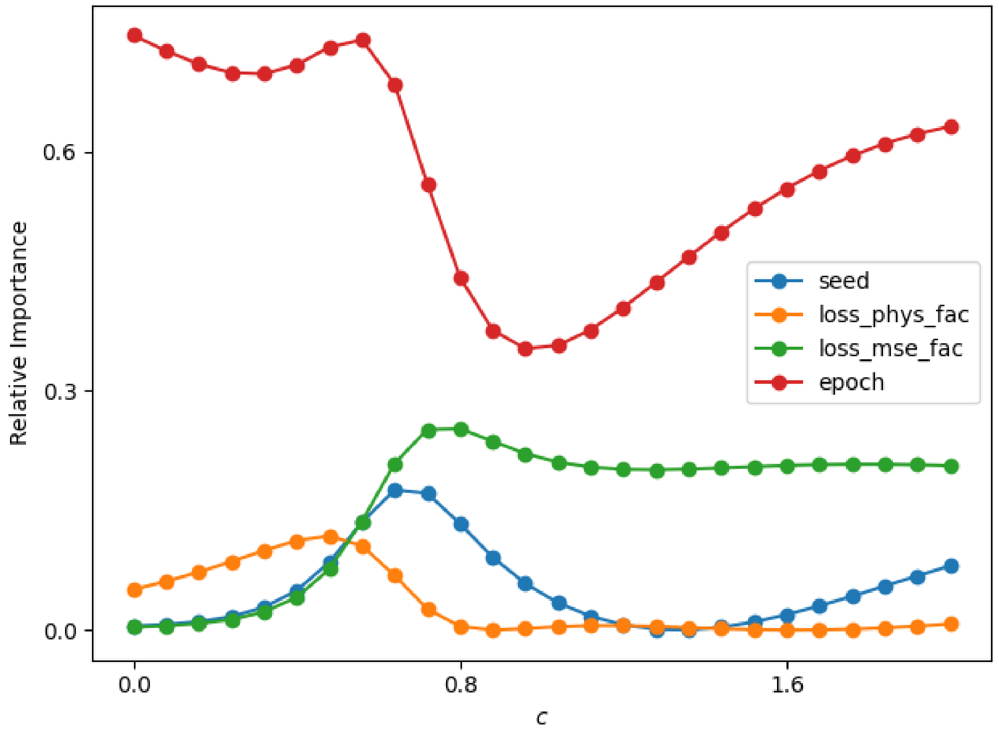

Figure 15: Relative importance on the variance of $R(c)$ of each hyperparameter for different values of c.

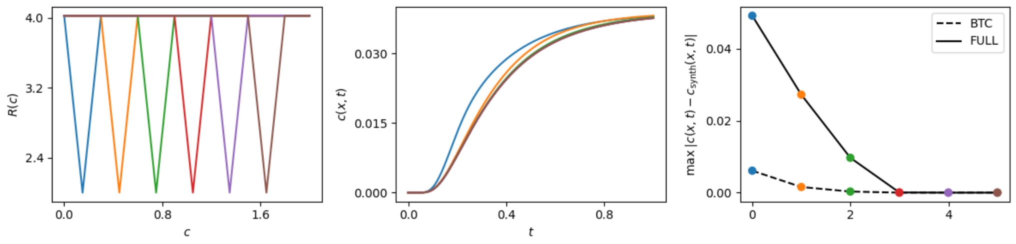

Figure 16: Left: Retardations with triangle peaks. Middle: Concentration BTC for these Retardations. Right: MAE for each on the full field and BTC.

8.2 Samples and PIs

The analysis of our results reveals several key insights regarding the performance and characteristics of the different uncertainty estimation methods, particularly focusing on the comparison between Full-SPAN and Markov Chain Monte Carlo (MCMC).

Firstly, the synthetic data case exhibits significantly lower uncertainty compared to the experimental data scenario. This can be attributed to several factors: a larger number of datapoints, which reduces epistemic uncertainty; the reduced effectiveness of Data-SPAN due to the absence of noise in the synthetic data; and the utilization of less effective SPAN hyperparameters (such as excluding the strongest options like "Max Epoch" or "MSE loss factor," as shown in Section 7.5).

Interestingly, Data-SPAN does not demonstrate a substantial improvement in uncertainty estimation with an increasing number of samples, especially when compared to MCMC. This is a favorable characteristic, as it implies that Data-SPAN can achieve reliable uncertainty estimates even with a limited number of samples and thus computational effort. Furthermore, SPAN benefits from trivial parallelizability, unlike MCMC, which requires sequential sampling or more advanced methods. This, coupled with Data-SPAN's ability to operate effectively with fewer samples, establishes SPAN as a more computationally efficient method for uncertainty quantification.

Although our sensitivity analysis indicated that certain hyperparameters exerted minimal influence on the results, their inclusion does not negatively impact performance. This is because Monte Carlo sampling, the core technique employed here, is inherently independent of dimensionality.

Finally, a comparison of the prediction intervals (PIs) generated by Full-SPAN and MCMC, specifically the $R(c)$ PI, and their respective likelihoods, reveals comparable performance. Qualitatively, both methods appear to achieve a similar level of accuracy in capturing uncertainty.

8.3 Limitations

One limitation of our approach lies in its "brute-force" nature, not only due to the computational effort but also due to the process itself. While we gain insights into uncertainty, the obtained result is highly dependent on the used solver and selected hyperparameters in the case of SPAN and highly depdendent on the given dataset in the case of Data-SPAN. Ideally, uncertainty is a byproduct of the method and the uncertainty in the data.

Furthermore, the weighting process raises concerns. The BTC is constrained to a small subset of concentration values compared to the full field (only 0 to 0.04 instead of 0 to 1.0). This raises questions about the validity of weighting e.g. $R(c = 0.0)$ samples the same way as $R(c = 1.0)$, as the latter do not directly influence the predicted BTC and thus their likelihood contribution is questionable.

9 Conclusion

9.1 Summary

This work investigated uncertainty quantification (UQ) for the retardation factor in a diffusion-sorption process using the FINN framework. We proposed two novel UQ methods, SPAN and Data-SPAN, based on perturbing hyperparameters and training data, respectively. Empirical analysis demonstrated the near-uniqueness of the inverse problem, enabling the interpretation of the

UQ results. While computationally more expensive than MCMC, our methods offer a different perspective on uncertainty by directly exploring the effects of various uncertainties within the training data and solver process.

9.2 Future Work

10 References

References

[1] Stefan Depeweg et al. “Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning”. In: *International conference on machine learning*. PMLR. 2018, pp. 1184–1193.

[2] Jakob Gawlikowski et al. “A survey of uncertainty in deep neural networks”. In: *Artificial Intelligence Review* 56.Suppl 1 (2023), pp. 1513–1589.

[3] Siyan Liu et al. “PI3NN: Out-of-distribution-aware prediction intervals from three neural networks”. In: *arXiv preprint arXiv:2108.02327* (2021).

[4] Wolfgang Nowak and Anneli Guthke. “Entropy-based experimental design for optimal model discrimination in the geosciences”. In: *Entropy* 18.11 (2016), p. 409.

[5] Timothy Praditia et al. “Learning groundwater contaminant diffusion-sorption processes with a finite volume neural network”. In: *Water Resources Research* 58.12 (2022), e2022WR033149.

[6] Ole Tange. *GNU Parallel 20231122 (’Grindavík’)*. GNU Parallel is a general parallelizer to run multiple serial command line programs in parallel without changing them. Nov. 2023. DOI: 10.5281/zenodo.10199085. URL: https://doi.org/10.5281/zenodo.10199085.