
Supplementary Material for “Uncertainty Quantification for Physics-Informed Neural Networks with Extended Fiducial Inference”

Frank Shih*

Department of Statistics
Purdue University
West Lafayette, IN 47907
zhan3761@purdue.edu

Zhenghao Jiang

Department of Statistics
Purdue University
West Lafayette, IN 47907
jiang976@purdue.edu

Faming Liang

Department of Statistics
Purdue University
West Lafayette, IN 47907
fmliang@purdue.edu

This supplementary material is organized as follows. Section A1 provides a brief review of the EFI method and its self-diagnostic property. Section A2 presents the analytical expression of the extended fiducial density (EFD) function of Z_n . Section A3 includes an illustrative plot of the proposed narrow-neck w -network. Section A4 gives the proof of Theorem 3.2. Section A5 presents additional numerical results. Finally, Section A6 details the parameter settings used in our numerical experiments.

A1 A Brief Review of the EFI Method and Its Self-Diagnosis

A1.1 The EFI Method

To ensure a smooth presentation of the EFI method, this section partially overlaps with Section 2 of the main text. Consider a regression model:

$$Y = f(\mathbf{X}, Z, \boldsymbol{\theta}),$$

where $Y \in \mathbb{R}$ and $\mathbf{X} \in \mathbb{R}^d$ represent the response and explanatory variables, respectively; $\boldsymbol{\theta} \in \mathbb{R}^p$ represents the vector of parameters; and $Z \in \mathbb{R}$ represents a scaled random error following a known distribution $\pi_0(\cdot)$. Suppose that a random sample of size n , denoted by $\{(y_1, \mathbf{x}_1), (y_2, \mathbf{x}_2), \dots, (y_n, \mathbf{x}_n)\}$, has been collected from the model. In structural inference, the observations can be expressed in data-generating equations as follows:

$$y_i = f(\mathbf{x}_i, z_i, \boldsymbol{\theta}), \quad i = 1, 2, \dots, n. \quad (\text{A1})$$

This system of equations consists of $n + p$ unknowns, namely, $\{\boldsymbol{\theta}, z_1, z_2, \dots, z_n\}$, while there are only n equations. Therefore, the values of $\boldsymbol{\theta}$ cannot be uniquely determined by the data-generating equations, and this lack of uniqueness of unknowns introduces uncertainty in $\boldsymbol{\theta}$.

Let $Z_n = \{z_1, z_2, \dots, z_n\}$ denote the unobservable random errors contained in the data, which are also called latent variables in EFI. Let $G(\cdot)$ denote an inverse function/mapping for $\boldsymbol{\theta}$, i.e.,

$$\boldsymbol{\theta} = G(\mathbf{Y}_n, \mathbf{X}_n, Z_n). \quad (\text{A2})$$

*Use footnote for providing further information about author (webpage, alternative address)—not for acknowledging funding agencies.

It is worth noting that the inverse function is generally non-unique. For example, it can be constructed by solving any p equations in (1) for θ . As noted by Liang et al. (2025), this non-uniqueness of inverse function mirrors the flexibility of frequentist methods, where different estimators of θ can be constructed to achieve desired properties such as efficiency, unbiasedness, and robustness.

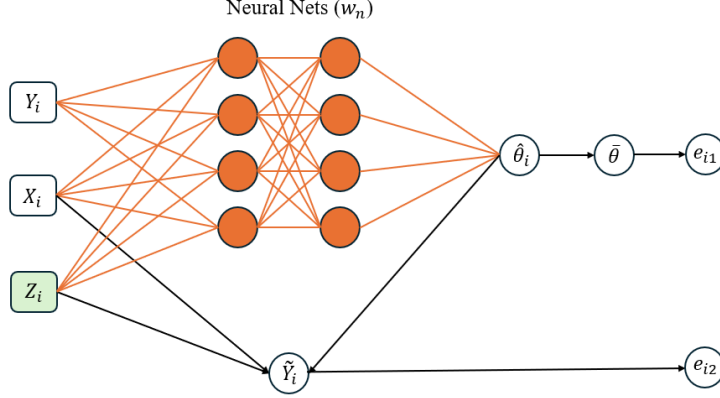


Figure A1: Diagram of EFI given in Liang et al. (2025): The orange nodes and orange links form a deep neural network (DNN), referred to as the w -network, which is parameterized by w_n (with the subscript n indicating its dependence on the training sample size n); the green node represents the latent variable to impute; and the black lines represent deterministic functions.

Since the inverse function $G(\cdot)$ is generally unknown, Liang et al. (2025) proposed to approximate it using a sparse DNN, see Figure A1 for illustration. The EFI network has two output nodes defined, respectively, by

$$e_{i,1} := \|\hat{\theta}_i - \bar{\theta}\|^2, \quad (A3)$$

and $e_{i,2} := d(y_i, \tilde{y}_i) := d(y_i, \mathbf{x}_i, z_i, \bar{\theta}),$

where $\tilde{y}_i = f(\mathbf{x}_i, z_i, \bar{\theta})$, $f(\cdot)$ is as specified in (1), and $d(\cdot)$ is a function measuring the difference between y_i and \tilde{y}_i . That is, the node $e_{i,1}$ quantifies the variation of $\hat{\theta}_i$, while the node $e_{i,2}$ represents the fitting error. For a normal linear/nonlinear regression, $d(\cdot)$ can be defined as

$$d(y_i, \mathbf{x}_i, z_i, \bar{\theta}) = \|y_i - f(\mathbf{x}_i, z_i, \bar{\theta})\|^2. \quad (A4)$$

For logistic regression, it is defined as a ReLU function, see Liang et al. (2025) for details.

Let $\hat{\theta}_i := \hat{g}(y_i, \mathbf{x}_i, z_i, w_n)$ denote the DNN prediction function parameterized by the weights w_n in the EFI network, and let

$$\bar{\theta} := \frac{1}{n} \sum_{i=1}^n \hat{\theta}_i = \frac{1}{n} \sum_{i=1}^n \hat{g}(y_i, \mathbf{x}_i, z_i, w_n), \quad (A5)$$

which serves as an estimator of the inverse function $G(\cdot)$.

EFI defines an energy function

$$U_n(\mathbf{Y}_n, \mathbf{X}_n, \mathbf{Z}_n, w_n) = \eta_\theta \sum_{i=1}^n \|\hat{\theta}_i - \bar{\theta}\|^2 + \sum_{i=1}^n d(y_i, \mathbf{x}_i, z_i, \bar{\theta}), \quad (A6)$$

where $\eta_\theta > 0$ is a regularization parameter, $\hat{\theta}_i$'s and $\bar{\theta}$ can be expressed as functions of $(\mathbf{Y}_n, \mathbf{X}_n, \mathbf{Z}_n, w_n)$, and $d(\cdot)$ is a function measuring the difference between y_i and \tilde{y}_i . The likelihood function is given by

$$\pi_\epsilon(\mathbf{Y}_n | \mathbf{X}_n, \mathbf{Z}_n, w_n) \propto e^{-U_n(\mathbf{Y}_n, \mathbf{X}_n, \mathbf{Z}_n, w_n)/\epsilon}, \quad (A7)$$

for some constant ϵ close to 0. As discussed in Liang et al. (2025), the choice of η_θ does not affect much on the performance of EFI as long as ϵ is sufficiently small. Subsequently, the posterior of w_n is given by

$$\pi_\epsilon(w_n | \mathbf{X}_n, \mathbf{Y}_n, \mathbf{Z}_n) \propto \pi(w_n) e^{-U_n(\mathbf{Y}_n, \mathbf{X}_n, \mathbf{Z}_n, w_n)/\epsilon},$$

where $\pi(\mathbf{w}_n)$ denotes the prior of \mathbf{w}_n ; and the predictive distribution of \mathbf{Z}_n is given by

$$\pi_\epsilon(\mathbf{Z}_n|\mathbf{X}_n, \mathbf{Y}_n, \mathbf{w}_n) \propto \pi_0^{\otimes n}(\mathbf{Z}_n)e^{-U_n(\mathbf{Y}_n, \mathbf{X}_n, \mathbf{Z}_n, \mathbf{w}_n)/\epsilon},$$

where $\pi_0^{\otimes n}(\mathbf{Z}_n) = \prod_{i=1}^n \pi_0(z_i)$ under the assumption that z_i 's are independently identically distributed (i.i.d.). In EFL, \mathbf{w}_n is estimated through maximizing the posterior $\pi_\epsilon(\mathbf{w}_n|\mathbf{X}_n, \mathbf{Y}_n)$ given the observations $\{\mathbf{X}_n, \mathbf{Y}_n\}$. By the Bayesian version of Fisher's identity (Song et al., 2020), the gradient equation $\nabla_{\mathbf{w}_n} \log \pi_\epsilon(\mathbf{w}_n|\mathbf{X}_n, \mathbf{Y}_n) = 0$ can be re-expressed as

$$\begin{aligned} \nabla_{\mathbf{w}_n} \log \pi_\epsilon(\mathbf{w}_n|\mathbf{X}_n, \mathbf{Y}_n) &= \int \left[\nabla_{\mathbf{w}_n} \log \pi_\epsilon(\mathbf{w}_n|\mathbf{X}_n, \mathbf{Y}_n, \mathbf{Z}_n) \right. \\ &\quad \left. \times \pi_\epsilon(\mathbf{Z}_n|\mathbf{X}_n, \mathbf{Y}_n, \mathbf{w}_n) \right] d\mathbf{Z}_n = 0, \end{aligned} \quad (\text{A8})$$

which can be solved using an adaptive stochastic gradient MCMC algorithm (Liang et al., 2022; Deng et al., 2019). The algorithm works by iterating between the *latent variable imputation* and *parameter updating* steps, see Algorithm 1 for the pseudo-code. This algorithm is termed "adaptive" because the transition kernel in the latent variable imputation step changes with the working parameter estimate of \mathbf{w}_n . The parameter updating step can be implemented using mini-batch SGD, and the latent variable imputation step can be executed in parallel for each observation (y_i, \mathbf{x}_i) . Hence, the algorithm is scalable with respect to large datasets.

Under mild conditions for the adaptive SGLD algorithm, it can be shown that

$$\|\mathbf{w}_n^{(k)} - \mathbf{w}_n^*\| \xrightarrow{P} 0, \quad \text{as } k \rightarrow \infty, \quad (\text{A9})$$

where \mathbf{w}_n^* denotes a solution to equation (3) and \xrightarrow{P} denotes convergence in probability, and that

$$\mathbf{Z}_n^{(k)} \overset{d}{\rightsquigarrow} \pi_\epsilon(\mathbf{Z}_n|\mathbf{X}_n, \mathbf{Y}_n, \mathbf{w}_n^*), \quad \text{as } k \rightarrow \infty, \quad (\text{A10})$$

in 2-Wasserstein distance, where $\overset{d}{\rightsquigarrow}$ denotes weak convergence. To study the limit of (6) as ϵ decays to 0, i.e.,

$$p_n^*(z|\mathbf{Y}_n, \mathbf{X}_n, \mathbf{w}_n^*) = \lim_{\epsilon \downarrow 0} \pi_\epsilon(\mathbf{Z}_n|\mathbf{X}_n, \mathbf{Y}_n, \mathbf{w}_n^*),$$

where $p_n^*(z|\mathbf{Y}_n, \mathbf{X}_n, \mathbf{w}_n^*)$ is referred to as the extended fiducial density (EFD) of \mathbf{Z}_n , Liang et al. (2025) impose specific conditions on the structure of the \mathbf{w} -network, including that the \mathbf{w} -network is sparse and that the output layer width (i.e., the dimension of $\boldsymbol{\theta}$) is either fixed or grows very slowly with the sample size n . Under these assumptions, they prove the consistency of \mathbf{w}_n^* based on the sparse deep learning theory developed in Sun et al. (2022). This consistency further implies that

$$G^*(\mathbf{Y}_n, \mathbf{X}_n, \mathbf{Z}_n) = \frac{1}{n} \sum_{i=1}^n \hat{g}(y_i, \mathbf{x}_i, z_i, \mathbf{w}_n^*), \quad (\text{A11})$$

serves as a consistent estimator for the inverse function/mapping $\boldsymbol{\theta} = G(\mathbf{Y}_n, \mathbf{X}_n, \mathbf{Z}_n)$. Refer to Appendix A2 for the analytic expression of $p_n^*(z|\mathbf{Y}_n, \mathbf{X}_n, \mathbf{w}_n^*)$.

Let $\mathcal{Z}_n = \{z \in \mathbb{R}^n : U_n(\mathbf{Y}_n, \mathbf{X}_n, \mathbf{Z}_n, \mathbf{w}_n^*) = 0\}$ denote the zero-energy set. Under some regularity conditions on the energy function, Liang et al. (2025) proved that \mathcal{Z}_n is invariant to the choice of $G(\cdot)$. Let $\Theta := \{\boldsymbol{\theta} \in \mathbb{R}^p : \boldsymbol{\theta} = G^*(\mathbf{Y}_n, \mathbf{X}_n, z), z \in \mathcal{Z}_n\}$ denote the parameter space of the target model, which represents the set of all possible values of $\boldsymbol{\theta}$ that $G^*(\cdot)$ takes when z runs over \mathcal{Z}_n . Then, for any function $b(\boldsymbol{\theta})$ of interest, its EFD $\mu_n^*(\cdot|\mathbf{Y}_n, \mathbf{X}_n)$ associated with $G^*(\cdot)$ is given by

$$\mu_n^*(B|\mathbf{Y}_n, \mathbf{X}_n) = \int_{\mathcal{Z}_n(B)} dP_n^*(z|\mathbf{Y}_n, \mathbf{X}_n, \mathbf{w}_n^*), \quad (\text{A12})$$

for any measurable set $B \subset \Theta$, where $\mathcal{Z}_n(B) = \{z \in \mathcal{Z}_n : b(G^*(\mathbf{Y}_n, \mathbf{X}_n, z)) \in B\}$, and $P_n^*(z|\mathbf{Y}_n, \mathbf{X}_n, \mathbf{w}_n^*)$ denote the cumulative distribution function (CDF) corresponding to $p_n^*(z|\mathbf{Y}_n, \mathbf{X}_n, \mathbf{w}_n^*)$. The EFD provides an uncertainty measure for $b(\boldsymbol{\theta})$. Practically, it can be constructed based on the samples $\{b(\boldsymbol{\theta}_1), b(\boldsymbol{\theta}_2), \dots, b(\boldsymbol{\theta}_M)\}$, where $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_M\}$ denotes the fiducial $\boldsymbol{\theta}$ -samples collected at step (iv) of Algorithm 1.

Finally, we note that for a neural network model, its parameters are only unique up to certain loss-invariant transformations, such as reordering hidden neurons within the same hidden layer or simultaneously altering the sign or scale of certain connection weights (Sun et al., 2022). Therefore, for the \mathbf{w} -network, the consistency of \mathbf{w}_n^* refers to its consistency with respect to one of the equivalent solutions to (3), while mathematically \mathbf{w}_n^* can still be treated as unique.

A1.2 Self-Diagnosis in EFI

Given the flexibility of DNN models, reliable diagnostics are crucial in deep learning to ensure model robustness and accuracy while identifying potential issues during training. Unlike dropout and Bayesian methods, which lack self-diagnostic capabilities, EFI includes a built-in mechanism for self-diagnosis. Specifically, this can be achieved through (i) analyzing the QQ-plot of the imputed random errors, and (ii) verifying that the energy function U_n converges to zero.

According to Lemma 3.1, the imputed random errors \hat{Z}_n should follow the same distribution as the true random errors Z_n . Since the theoretical distribution of Z_n is known, the convergence of \hat{Z}_n can be assessed using QQ-plot as shown in Figure 2(b). When Z_n has been correctly imputed, the energy function must converge to zero to ensure the consistency of the inverse function estimator. In practice, we can check whether $U_n(Y_n, X_n, Z_n, w_n) = o(\epsilon)$ as $\epsilon \rightarrow 0$. The validity of inference for the model uncertainty can thus be ensured if both diagnostic tests are satisfied. This diagnostic method is entirely data-driven, offering a simple way for validating the EFI results.

If the diagnostic tests are not satisfied, the hyperparameters of EFI can be adjusted to ensure both tests are met for valid inference. These adjustments may include modifying the width of the neck layer, the size of the $w_n^{(1)}$ -network, the size of the data-modeling network, as well as tuning the learning rates and iteration numbers used in Algorithm 1.

A2 Extended Fiducial Density Function of Z_n

Let $\mathcal{Z}_n = \{z \in \mathbb{R}^n : U_n(Y_n, X_n, Z_n, w_n^*) = 0\}$ denote the zero-energy set, and let $P_n^*(z|X_n, Y_n, w_n^*)$ denote the cumulative distribution function (CDF) corresponding to $p_n^*(z|X_n, Y_n, w_n^*)$. Under some regularity conditions on the energy function, Liang et al. (2025) proved that \mathcal{Z}_n is invariant to the choice of $G(\cdot)$. Furthermore, they studied the convergence of $\lim_{\epsilon \downarrow 0} \pi_\epsilon(z|X_n, Y_n, w_n^*)$ in two cases: $\Pi_n(\mathcal{Z}_n) > 0$ and $\Pi_n(\mathcal{Z}_n) = 0$, where $\Pi_n(\cdot)$ denotes the probability measure corresponding to the density function $\pi_0^{\otimes n}(z)$ on \mathbb{R}^n . Specifically,

(a) ($\Pi_n(\mathcal{Z}_n) > 0$): In this case, $p_n^*(z|X_n, Y_n, w_n^*)$ is given by

$$\frac{dP_n^*(z|X_n, Y_n, w_n^*)}{dz} = \frac{1}{\Pi_n(\mathcal{Z}_n)} \pi_0^{\otimes n}(z), \quad z \in \mathcal{Z}_n, \quad (\text{A13})$$

which is invariant to the choices of the inverse function $G(\cdot)$ and energy function $U_n(\cdot)$. For example, the logistic regression belongs to this case as shown in Liang et al. (2025).

(b) ($\Pi_n(\mathcal{Z}_n) = 0$): In this case, \mathcal{Z}_n forms a manifold in \mathbb{R}^n with the highest dimension p , and $p_n^*(z|X_n, Y_n, w_n^*)$ concentrates on the highest dimensional manifold and is given by

$$\frac{dP_n^*(z|X_n, Y_n, w_n^*)}{d\nu}(z) = \frac{\pi_0^{\otimes n}(z) (\det(\nabla_t^2 U_n(z)))^{-1/2}}{\int_{\mathcal{Z}_n} \pi_0^{\otimes n}(z) (\det(\nabla_t^2 U_n(z)))^{-1/2} d\nu}, \quad z \in \mathcal{Z}_n, \quad (\text{A14})$$

where ν is the sum of intrinsic measures on the p -dimensional manifold in \mathcal{Z}_n , and $t \in \mathbb{R}^{n-p}$ denotes the coefficients of the normalized smooth normal vectors in the tubular neighborhood decomposition (Milnor and Stasheff, 1974) of z .

By Theorem 3.2 and Lemma 4.2 in Liang et al. (2025), if the target model is noise-additive and $d(\cdot)$ is specified as in (A4), then $P_n^*(z|X_n, Y_n, w_n^*)$ in (A14) can be reduced to

$$\frac{dP_n^*(z|X_n, Y_n, w_n^*)}{d\nu} = \frac{\pi_0^{\otimes n}(z)}{\int_{\mathcal{Z}_n} \pi_0^{\otimes n}(z) d\nu}. \quad (\text{A15})$$

That is, under the consistency of w_n^* , $p_n^*(z|X_n, Y_n, w_n^*)$ is reduced to a truncated density function of $\pi_0^{\otimes n}(z)$ on the manifold \mathcal{Z}_n , while \mathcal{Z}_n itself is also invariant to the choice of the inverse function. In other words, for noise-additive models, the EFD of Z_n is asymptotically invariant to the inverse function we learned given its consistency.

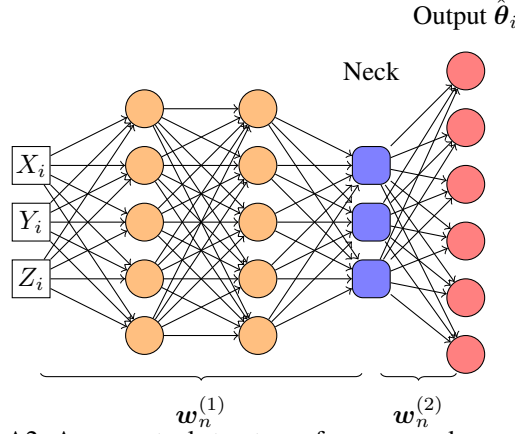


Figure A2: A conceptual structure of narrow neck w -networks.

A3 Narrow-Neck w -Networks

A4 Proof of Theorem 3.2

A4.1 Asymptotic Equivalence between StoNets and DNNs

We first provide a brief review of the theory regarding asymptotic equivalence between the StoNet and DNN models, which was originally established in [Liang et al. \(2022\)](#).

Consider a StoNet model:

$$\begin{aligned} Y_1 &= b_1 + w_1 X + e_1, \\ Y_i &= b_i + w_i \Psi(Y_{i-1}) + e_i, \quad i = 2, 3, \dots, h, \\ Y &= b_{h+1} + w_{h+1} \Psi(Y_h) + e_{h+1}, \end{aligned} \quad (\text{A16})$$

where $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}^{d_{h+1}}$ represent the input and response variables, respectively; $Y_i \in \mathbb{R}^{d_i}$ are latent variables; $e_i \in \mathbb{R}^{d_i}$ are introduced noise variables; $b_i \in \mathbb{R}^{d_i}$ and $w_i \in \mathbb{R}^{d_i \times d_{i-1}}$ are model parameters, $d_0 = p$, and $\Psi(Y_{i-1}) = (\psi(Y_{i-1,1}), \psi(Y_{i-1,2}), \dots, \psi(Y_{i-1,d_{i-1}}))^T$ is an element-wise activation function for $i = 1, \dots, h+1$. The StoNet defines a latent variable model that reformulates the DNN as a composition of many simple regressions. In this context, we assume that $e_i \sim N(0, \sigma_i^2 I_{d_i})$ for $i = 1, 2, \dots, h, h+1$, though other distributions can also be considered for e_i 's ([Sun and Liang, 2022](#)).

The DNN model corresponding to (A16) is given as follows:

$$\begin{aligned} \tilde{Y}_1 &= b_1 + w_1 X, \\ \tilde{Y}_i &= b_i + w_i \Psi(\tilde{Y}_{i-1}), \quad i = 2, 3, \dots, h, \\ Y &= b_{h+1} + w_{h+1} \Psi(\tilde{Y}_h) + e_{h+1}. \end{aligned} \quad (\text{A17})$$

Let $\vartheta = \{b_1, w_1, \dots, b_{h+1}, w_{h+1}\}$ denote the parameter set of the DNN model. Let $\pi_{\text{DNN}}(Y|X, \vartheta)$ denote the likelihood function of the DNN model (A17), and let $\pi(Y, Y_{\text{mis}}|X, \vartheta)$ denote the likelihood function of the StoNet (A16). Let $Q^*(\vartheta) = \mathbb{E}(\log \pi_{\text{DNN}}(Y|X, \vartheta))$, where the expectation is taken with respect to the joint distribution $\pi(X, Y)$, [Liang et al. \(2022\)](#) made the following assumption regarding the network structure, activation function, and the variance of the latent variables of StoNet:

Assumption A4.1. (i) Parameter space $\tilde{\Theta}$ (of ϑ) is compact; (ii) For any $\vartheta \in \tilde{\Theta}$, $\mathbb{E}(\log \pi(Y, Y_{\text{mis}}|X, \vartheta))^2 < \infty$; (iii) The activation function $\psi(\cdot)$ is c -Lipschitz continuous; (iv) The network's widths d_i 's and depth h are allowed to increase with n ; (v) The noise introduced in StoNet satisfies the following condition: $\sigma_1 \leq \sigma_2 \leq \dots \leq \sigma_{h+1}$, and $d_{h+1}(\prod_{i=k+1}^h d_i^2) d_k \sigma_k^2 \prec \frac{\sigma_{h+1}^2}{h}$ for any $k \in \{1, 2, \dots, h\}$.

Assumption A4.2. (i) $Q^*(\vartheta)$ is continuous in ϑ and uniquely maximized at ϑ^* ; (ii) for any $\epsilon > 0$, $\sup_{\vartheta \in \Theta \setminus B(\epsilon)} Q^*(\vartheta)$ exists, where $B(\epsilon) = \{\vartheta : \|\vartheta - \vartheta^*\| < \epsilon\}$, and $\delta = Q^*(\vartheta^*) - \sup_{\vartheta \in \Theta \setminus B(\epsilon)} Q^*(\vartheta) > 0$.

Under Assumptions A4.1 and A4.2, Liang et al. (2022) proved the following lemma.

Lemma A4.3. (Liang et al., 2022) Suppose that Assumptions A4.1-A4.2 hold, and $\pi(\mathbf{Y}, \mathbf{Y}_{\text{mis}}|\mathbf{X}, \boldsymbol{\vartheta})$ is continuous in $\boldsymbol{\vartheta}$. Then

$$(i) \quad \sup_{\boldsymbol{\vartheta} \in \Theta} \left| \frac{1}{n} \sum_{i=1}^n \log \pi(\mathbf{Y}^{(i)}, \mathbf{Y}_{\text{mis}}^{(i)}|\mathbf{X}^{(i)}, \boldsymbol{\vartheta}) - \frac{1}{n} \sum_{i=1}^n \log \pi_{\text{DNN}}(\mathbf{Y}^{(i)}|\mathbf{X}^{(i)}, \boldsymbol{\vartheta}) \right| \xrightarrow{P} 0, \quad (A18)$$

$$(ii) \quad \|\hat{\boldsymbol{\vartheta}}_n - \boldsymbol{\vartheta}^*\| \xrightarrow{P} 0, \quad \text{as } n \rightarrow \infty,$$

where $\boldsymbol{\vartheta}^* = \arg \max_{\boldsymbol{\vartheta} \in \Theta} \mathbb{E}(\log \pi_{\text{DNN}}(\mathbf{Y}|\mathbf{X}, \boldsymbol{\vartheta}))$ denotes the true parameters of the DNN model as specified in (A17), and $\hat{\boldsymbol{\vartheta}}_n = \arg \max_{\boldsymbol{\vartheta} \in \Theta} \{\frac{1}{n} \sum_{i=1}^n \log \pi(\mathbf{Y}^{(i)}, \mathbf{Y}_{\text{mis}}^{(i)}|\mathbf{X}^{(i)}, \boldsymbol{\vartheta})\}$ denotes the maximum likelihood estimator of the StoNet model (A16) with the pseudo-complete data.

Lemma A4.3 implies that the StoNet and DNN are asymptotically equivalent as the training sample size n becomes large, and it forms the basis for the bridging the StoNet and the DNN. The asymptotic equivalence can be elaborated from two perspectives. First, suppose the DNN model (A17) is true. Lemma A4.3 implies that when n becomes large, the weights of the DNN can be learned by training a StoNet of the same structure with σ_i^2 's satisfying Assumption A4.1-(v). On the other hand, suppose that the StoNet (A16) is true, and then Lemma A4.3 implies that for any StoNet satisfying Assumptions A4.1 & A4.2, the weights $\boldsymbol{\vartheta}$ can be learned by training a DNN of the same structure when the training sample size is large.

A4.2 Justification for Condition (12)-(ii)

To justify this condition, we first introduce the following lemma:

Lemma A4.4. Consider a random matrix $\mathbb{M} \in \mathbb{R}^{n \times d}$ with $n \geq d$. Suppose that the eigenvalues of $\mathbb{M}^T \mathbb{M}$ are upper bounded, i.e., $\lambda_{\max}(\mathbb{M}^T \mathbb{M}) \leq \kappa_{\max}$ for some constant $\kappa_{\max} > 0$. Let $\Psi(\mathbb{M})$ denote an elementwise transformation of \mathbb{M} . Then $\lambda_{\max}((\Psi(\mathbb{M}))^T (\Psi(\mathbb{M}))) \leq \kappa_{\max}$ for the *tanh*, *sigmoid* and *ReLU* transformations.

Proof. For ReLU, the result follows from Lemma 5 of Dittmer et al. (2018). For *tanh* and *sigmoid*, since they are Lipschitz continuous with a Lipschitz constant of 1, Lemma 5 of Dittmer et al. (2018) also applies. \square

Since the connection weights take values in a compact space $\tilde{\Theta}$, there exists a constant $0 < \tau_{\max} < \infty$ such that

$$\lambda_{\max}(\mathbf{w}_l \mathbf{w}_l^T) \leq \tau_{\max},$$

for any $l = 1, 2, \dots, h+1$, where $\mathbf{w}_l \in \mathbb{R}^{d_l \times d_{l-1}}$ is the weight matrix of the DNN at layer l .

Let $\mathbb{M}_l \in \mathbb{R}^{n \times d_l}$ denote the output of hidden layer $l \in \{1, 2, \dots, h\}$ of the StoNet; that is,

$$\begin{aligned} \mathbb{M}_l &= \Psi(\Psi(\mathbb{M}_{l-1}) \mathbf{w}_l^T), \quad l = 1, 2, \dots, h-1, \\ \mathbb{M}_h &= \Psi(\Psi(\mathbb{M}_{h-1}) \mathbf{w}_h^T + \mathbb{V}). \end{aligned} \quad (A19)$$

Consider the case that the activation functions are bounded, such as *sigmoid* and *tanh*. Then the matrix $\mathbb{E}(\mathbb{M}_h^T \mathbb{M}_h)$ has the eigenvalues upper bounded by $n\kappa_{\max}$ for some constant $0 < \kappa_{\max} < \infty$.

For the case that the activation functions are unbounded, such as *ReLU* or *leaky ReLU*, we can employ the layer-normalization method in training. In this case, by Lemma A4.4, the matrix $\Psi(\mathbb{M}_{h-1})^T \Psi(\mathbb{M}_{h-1})$ has the eigenvalues upper bounded by $n\kappa_{\max}$, provided that the eigenvalues of the matrix $\mathbb{M}_{h-1}^T \mathbb{M}_{h-1}$ is bounded by $n\kappa_{\max}$ after layer-normalization.

Let $\tilde{\mathbb{M}}_h = \Psi(\mathbb{M}_{h-1}) \mathbf{w}_h^T$. Then, by an extension of Ostrowski's theorem, see Theorem 3.2 of Higham and Cheng (1998), the eigenvalues of the matrix $\tilde{\mathbb{M}}_h^T \tilde{\mathbb{M}}_h = \mathbf{w}_h (\Psi(\mathbb{M}_{h-1}))^T \Psi(\mathbb{M}_{h-1}) \mathbf{w}_h^T$ is bounded by

$$\lambda_{\max}(\tilde{\mathbb{M}}_h^T \tilde{\mathbb{M}}_h) \leq n\kappa_{\max} \tau_{\max}.$$

Also, we have $\frac{1}{n} \lambda_{\max}(\mathbb{E}(\mathbb{V}^T \mathbb{V})) = \sigma_v^2$ under the assumption (*). Then, with the use of Lemma A4.4 and the Cauchy-Schwarz inequality, we have

$$\frac{1}{n} \lambda_{\max}(\mathbb{E}(\mathbb{M}_h^T \mathbb{M}_h)) \leq \kappa_{\max} \tau_{\max} + \sigma_v^2 + 2\sigma_v \sqrt{\kappa_{\max} \tau_{\max}},$$

as the sample size n becomes large. This concludes the proof for condition (12)-(ii).

A4.3 Proof of Theorem 3.2

Proof. First, we note that σ_θ^2 , the variance of each component of $\hat{\theta}_i (\in \mathbb{R}^p)$, is of the order $O(\epsilon/\eta)$ under the setting of the energy function (A6). By setting $\sigma_u^2 \prec \frac{\epsilon}{\eta h d_h p}$, it is easy to verify that the w -network satisfies Assumption A4.1. Therefore, under the additional regularity conditions given in Assumption A4.2, the proposed stochastic w -network is asymptotically equivalent to the original w -network. Furthermore, the stochastic w -network can be trained by training the original w -network using Algorithm 1.

Next, we prove that the inverse mapping learned through training the stochastic w -network is consistent. For Lemma 3.1, Liang et al. (2025) first established the convergence of the weights to w_n^* , and subsequently established the weak convergence of the latent variables. Therefore, due to the Markov chain nature of Algorithm 1, it suffices to prove that the inverse mapping produced by the StoNet is consistent, assuming the latent variables have been correctly imputed (i.e., the true values of Z_n are known). Once the inverse mapping's consistency is established, the latent variables will be correctly imputed in the next iteration, owing to the algorithm's equation-solving nature, which is ensured by setting $\epsilon \rightarrow 0$ upon convergence to the zero-energy region. This ensures that algorithm remains in its equilibrium, enabling accurate inference of model uncertainty.

Under the StoNet setting, estimation of $w_n^{(2)}$ involves solving p low-dimensional regressions. In particular, given $\mathbb{M} = (\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_n)^T \in \mathbb{R}^{n \times d_l}$, solving each of the regressions contributes a parameter estimation error:

$$\mathbb{E} \|\hat{\xi}_j^m - \xi_j^*\|^2 = \frac{\sigma_\theta^2}{n} \text{Tr}([\mathbb{E}(\mathbf{m}_i \mathbf{m}_i^T)]^{-1}) \leq \frac{d_h \sigma_\theta^2}{n \rho_{\min}} = O\left(\frac{d_h \epsilon}{n \eta}\right), \quad (\text{A20})$$

where ξ_i^* denotes the i th column of $w_n^{(2)}$, i.e., $w_n^{(2)} := (\xi_1^*, \dots, \xi_p^*)^T \in \mathbb{R}^{p \times d_h}$; $\hat{\xi}_j^m = (\mathbb{M}^T \mathbb{M})^{-1} \mathbb{M}^T \hat{\theta}^{(j)}$ is the OLS estimator for the regression coefficients; and $\hat{\theta}^{(j)} = (\hat{\theta}_1^{(j)}, \dots, \hat{\theta}_n^{(j)})^T$.

Let $\hat{w}^{m(2)} = (\hat{\xi}_1^m, \dots, \hat{\xi}_p^m)^T \in \mathbb{R}^{p \times d_h}$. By a fundamental property of linear regression, the mean prediction $\hat{\theta}_i^* = \hat{w}^{m(2)} \mathbf{m}_i$ is consistent with respect to $\theta^* = \mathbb{E}(\hat{\theta}_i)$, provided that $d_h \prec n$, the \mathbf{m}_i 's extract all θ -relevant information from the input variables, and the w -network has sufficient capacity to establish the linear relationship $\hat{\theta}_i \sim \mathbf{m}_i$ for $i = 1, 2, \dots, n$. The equality $\theta^* = \mathbb{E}(\hat{\theta}_i)$ is guaranteed by the setting of $\epsilon \rightarrow 0$ and by the construction of the energy function, which approaches to zero if and only if the empirical mean $\frac{1}{n} \sum_{i=1}^n \hat{\theta}_i$ converges to θ^* and the variance of $\hat{\theta}_i$ approaches to zero. Furthermore, as shown in Liang et al. (2022), the stochastic layer effectively provides a sufficient dimension reduction for the input variables.

Finally, we prove that the inverse mapping obtained by Algorithm 1 during the training of the non-stochastic w -network is also consistent. Let $\tilde{\theta}_i^* = w^{(2)} \Psi(\mu_i)$. Let $\tilde{\theta}_i^{*(j)}$ and $\hat{\theta}_i^{*(j)}$ denote the j th element of $\tilde{\theta}_i^*$ and $\hat{\theta}_i^*$, respectively. From equation (*), we have

$$\begin{aligned} \mathbb{E} |\hat{\theta}_i^{*(j)} - \tilde{\theta}_i^{*(j)}| &= \mathbb{E} |(\hat{\xi}_j^m)^T \mathbf{m}_i - (\xi_j^*)^T \Psi(\mu_i)| \\ &\leq \mathbb{E} |(\hat{\xi}_j^m - \xi_j^*)^T \mathbf{m}_i| + \mathbb{E} |(\xi_j^*)^T (\mathbf{m}_i - \Psi(\mu_i))| \\ &\leq (\mathbb{E} \|\hat{\xi}_j^m - \xi_j^*\|^2)^{1/2} (\mathbb{E} \|\mathbf{m}_i\|^2)^{1/2} + (\mathbb{E} \|\xi_j^*\|^2)^{1/2} (\mathbb{E} \|\mathbf{m}_i - \Psi(\mu_i)\|^2)^{1/2} \\ &\leq (\mathbb{E} \|\hat{\xi}_j^m - \xi_j^*\|^2)^{1/2} (\text{Tr}(\mathbb{E}(\mathbf{m}_i \mathbf{m}_i^T)))^{1/2} + c(\mathbb{E} \|\xi_j^*\|^2)^{1/2} (\text{Tr}(\sigma_v^2 I_{d_h}))^{1/2} \\ &\lesssim \sqrt{\frac{d_h \epsilon}{n \eta}} \sqrt{d_h \rho_{\max}} + c d_{\tilde{\Theta}} d_h \sigma_v, \end{aligned} \quad (\text{A21})$$

where $d_{\tilde{\Theta}}$ denotes the radius of the parameter space $\tilde{\Theta}$ (centered at 0), the second inequality follows from Cauchy-Schwarz inequality, the third inequality follows from the Taylor expansion for $\Psi(\mu_i + v_i)$ (at the point μ_i), and the last inequality follows from (A20), condition (12)-(ii), and the boundedness of ξ_j^* as stated in Assumption A4.1-(i).

Substituting $\sigma_v \prec \sqrt{\frac{\epsilon}{\eta h d_h p}}$ and ignoring some constant factors in (A21), we obtain

$$\mathbb{E} \|\hat{\theta}_i^* - \tilde{\theta}_i^*\|_1 \leq p \mathbb{E} |\hat{\theta}_i^{*(j)} - \tilde{\theta}_i^{*(j)}| \lesssim p d_h \sqrt{\frac{\epsilon}{n}} + \sqrt{\frac{\epsilon p d_h}{h}},$$

where $\|\cdot\|_1$ denotes the l_1 -norm of a vector.

By setting $\epsilon \prec \min\{\frac{n}{p^2 d_h^2}, \frac{h}{p d_h}\}$, we have $\mathbb{E} \|\hat{\theta}_i^* - \tilde{\theta}_i^*\|_1 = o(1)$, which implies $\tilde{\theta}_i^*$ is also consistent with respect to $\theta^* = \mathbb{E}(\hat{\theta}_i)$. Consequently, the inverse mapping $\frac{1}{n} \sum_{i=1}^n \tilde{\theta}_i^*$ produced by Algorithm 1 in training the non-stochastic w -network is also consistent with respect to θ^* . This concludes the proof of the theorem. \square

A5 Additional Numerical Results

Regarding uncertainty quantification, we note that there are two types of uncertainties:

1. **Aleatoric uncertainty:** This refers to the *irreducible noise* inherent in the data-generating process. It can be modeled as

$$y_i = f(x_i | \theta) + \epsilon_i,$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Estimating the unknown variance σ^2 corresponds to quantifying the aleatoric uncertainty (system random error). This is precisely what we addressed in the last experiment included in our previous rebuttal, where $\sigma^2 = 0.1$ was treated as unknown.

2. **Epistemic uncertainty:** This refers to the *reducible estimation error* due to limited data or incomplete knowledge of the true model (see, for example, model comparison in Section A5.8). In classical statistics, confidence intervals quantify epistemic uncertainty: as the dataset size increases, epistemic uncertainty—and thus the width of the confidence interval—decreases.

In the following section, we demonstrate through different examples that EFI is able to accurately quantify both types of uncertainties. We use the coverage rate as the key metric to quantify epistemic uncertainty. The coverage rate can reach the nominal level only when the parameter estimates are unbiased and the uncertainty estimation is accurate.

A5.1 1-D Poisson Equation

Figure A3 and Figure A4 provide typical trajectories learned for the 1-D Poisson model (13) using the methods: PINN, Dropout, and Bayesian PINN. For the ablation study, we vary the noise standard deviation from 0.01 to 0.1 to further assess the validity and accuracy of the proposed method in uncertainty quantification. The results, presented in Tables A1, A2, and A3, show that under different noise levels, the EFI algorithm consistently achieves a 95% coverage rate for the 95% confidence intervals.

To further investigate the relationship between confidence intervals and epistemic uncertainty, we conducted an additional experiment using two different sample sizes: 20 and 80. The results, presented in Table A4, clearly show that as the sample size increases, the width of the confidence interval decreases while the coverage rate remains consistent. This demonstrates that the confidence interval effectively captures the reducible nature of epistemic uncertainty.

A5.2 1-D Poisson Equation with f -measurement error

We revisit the same 1-D Poisson model as defined in (13), but now incorporate measurement errors in both u and f . Specifically, we consider 4 sensors for u and 40 sensors for f , with each sensor recording 10 replicate measurements. Measurement errors are modeled as $z_i^u \sim N(0, 0.05^2)$ and $z_i^f \sim N(0, 0.05^2)$, and we simulate 100 independent datasets under this setting. The experimental results are summarized in Table A5.

For the Bayesian PINN (B-PINN) method (Yang et al., 2021), we set $\sigma_u = \sigma_f = 0.05$, ensuring the likelihood function is correctly specified. However, as shown in Table A5, B-PINN produces

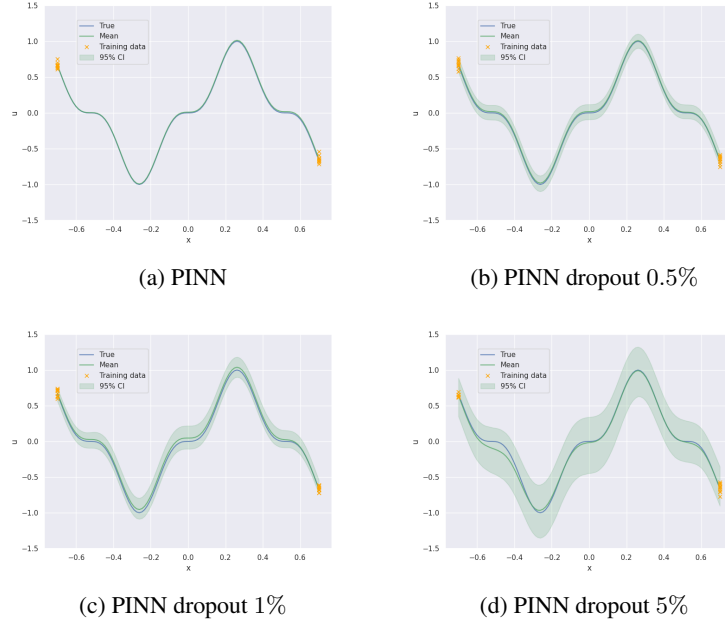


Figure A3: 1-D Poisson model (13): (a) Trajectory learned by PINN (without uncertainties); (b)-(d) Trajectories learned by Dropout with different dropout rates.

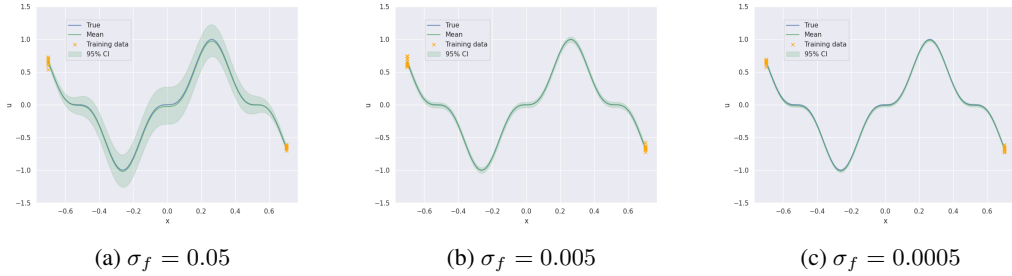


Figure A4: 1-D Poisson model (13): Typical trajectories learned by Bayesian PINN with different σ_f values.

excessively wide confidence intervals, resulting in an inflated and inaccurate coverage rate. This highlights another issue inherent to Bayesian DNNs, as noted in [Liang et al. \(2018\)](#); [Sun et al. \(2022\)](#): their performance can be significantly affected by the choice of prior in small- n -large- p settings. Here, p refers to the number of parameters in the DNN used to approximate the solution $u(x)$. Similarly, the dropout method continues to produce overly wide confidence intervals and inflated coverage rates.

In contrast, EFI achieves a coverage rate of 94.88% with the smallest confidence interval width. Notably, this experiment involves noise in both u and f observations. To evaluate the imputed errors, the QQ-plot of \hat{z}_i^u and \hat{z}_i^f across 100 experiments is shown in Figure A5. The Q-Q plot confirms that the distribution of imputed errors closely follows its theoretical distribution, supporting the validity of the EFI approach in handling measurement noise from different sources.

A5.3 Computing Time

Table A6 reports the wall-clock time for the Poisson-1D experiment with different algorithms. As shown by the table, EFI is slower than PINN (with dropout) but much faster than B-PINN (with the HMC sampler). Importantly, we have demonstrated that EFI is the only existing method capable of correctly constructing confidence intervals in a statistically rigorous manner. For larger networks,

Table A1: Comparison of different methods for 1D-Poisson with $\sigma = 0.01$

Method	MSE	Coverage Rate	CI-Width
PINN (no dropout)	0.000008 (0.000001)	0.2808 (0.028721)	0.002079 (0.000026)
Dropout (0.5%)	0.000160 (0.000037)	1.0000 (0.000000)	0.199798 (0.004135)
Dropout (1%)	0.000676 (0.000082)	1.0000 (0.000000)	0.276534 (0.006642)
Dropout (5%)	0.009080 (0.001516)	0.9639 (0.011218)	0.593541 (0.025812)
Bayesian ($\sigma_f = 0.05$)	0.000181 (0.000025)	0.9984 (0.000400)	0.254861 (0.001554)
Bayesian ($\sigma_f = 0.005$)	0.000007 (0.000001)	0.9915 (0.003220)	0.028386 (0.000201)
Bayesian ($\sigma_f = 0.0005$)	0.000007 (0.000001)	0.9544 (0.013751)	0.010437 (0.000046)
EFI	0.000007 (0.000001)	0.9423 (0.015229)	0.010759 (0.000117)

Table A2: Comparison of different methods for 1D-Poisson with $\sigma = 0.025$

Method	MSE	Coverage Rate	CI-Width
PINN (no dropout)	0.000046 (0.000006)	0.1529 (0.019175)	0.002110 (0.000025)
Dropout (0.5%)	0.000157 (0.000028)	1.0000 (0.000000)	0.195658 (0.003892)
Dropout (1%)	0.000666 (0.000084)	1.0000 (0.000000)	0.267936 (0.005751)
Dropout (5%)	0.004573 (0.000654)	0.9979 (0.001157)	0.643227 (0.031514)
Bayesian ($\sigma_f = 0.05$)	0.000161 (0.000019)	0.9992 (0.000307)	0.257184 (0.001612)
Bayesian ($\sigma_f = 0.005$)	0.000060 (0.000013)	0.9630 (0.012690)	0.037341 (0.000418)
Bayesian ($\sigma_f = 0.0005$)	0.000090 (0.000014)	0.8504 (0.023270)	0.024676 (0.000366)
EFI	0.000048 (0.000006)	0.9577 (0.014487)	0.027845 (0.000115)

Table A3: Comparison of different methods for 1D-Poisson with $\sigma = 0.1$

Method	MSE	Coverage Rate	CI-Width
PINN (no dropout)	0.000581 (0.000066)	0.0320 (0.006518)	0.002088 (0.000024)
Dropout (0.5%)	0.000935 (0.000106)	0.9905 (0.002706)	0.196226 (0.003748)
Dropout (1%)	0.001413 (0.000175)	0.9976 (0.001929)	0.274188 (0.007653)
Dropout (5%)	0.006840 (0.001027)	0.9955 (0.002439)	0.658607 (0.031770)
Bayesian ($\sigma_f = 0.05$)	0.000966 (0.000106)	0.9936 (0.002068)	0.285660 (0.001405)
Bayesian ($\sigma_f = 0.005$)	0.000674 (0.000080)	0.9582 (0.015014)	0.104743 (0.000325)
Bayesian ($\sigma_f = 0.0005$)	0.004944 (0.000788)	0.3827 (0.030245)	0.066296 (0.002515)
EFI	0.000624 (0.000058)	0.9493 (0.013526)	0.099543 (0.000775)

(a) QQ-plot of imputed errors in u (b) QQ-plot of imputed errors in f (c) Confidence intervalFigure A5: EFI-PINN diagnostic for the 1-D Poisson model (13) (with f -measurement error)

Table A4: Numerical results for 1D-Poisson with $\sigma = 0.1$, where the confidence interval width shrinks as sample size increases. The results are computed based on 100 independent simulations.

Method	n_b (sample size)	MSE	Coverage Rate	CI-Width
EFI	20	0.000624 (0.000058)	0.9493 (0.013526)	0.099543 (0.000775)
EFI	80	0.000173 (0.000017)	0.9501 (0.013459)	0.054256 (0.000723)

Table A5: Comparison of different methods for the 1-D Poisson model (13) (with f -measurement error), averaged over 100 runs.

Method	hidden layers	MSE	Coverage Rate	CI-Width
PINN	[50, 50]	0.000271 (0.000019)	0.0921 (0.008076)	0.003625 (0.000085)
Dropout (0.5%)	[50, 50]	0.000310 (0.000022)	1.0000 (0.000000)	0.240585 (0.002282)
Dropout (1.0%)	[50, 50]	0.000530 (0.000046)	1.0000 (0.000000)	0.357782 (0.005141)
Dropout (5.0%)	[50, 50]	0.003527 (0.000193)	1.0000 (0.000000)	0.669799 (0.006158)
Bayesian	[50, 50]	0.000233 (0.000017)	0.9960 (0.002035)	0.086291 (0.000068)
EFI	[50, 50]	0.000238 (0.000017)	0.9488 (0.009419)	0.060269 (0.000365)

we can adopt transfer learning techniques by constructing EFI hyper-networks only for the last few layers of the θ -network, which would significantly reduce the computational cost.

Table A6: Wall-clock time for PINN, B-PINN and EFI-PINN

Algorithm	Hypernetwork	Epoch ($\times 10^3$)	Wall-clock time (s)	Time per epoch (ms)
PINN (dropout)	-	200	133	0.665
B-PINN	-	100	1330	13.300
EFI	[16,16,16]	200	391	1.955
EFI	[16,16,4]	200	385	1.925

A5.4 Non-linear Poisson Equation

We extend our study to a non-linear Poisson equation given by:

$$\beta \frac{\partial^2 u}{\partial x^2} + k \tanh(u) = f, \quad x \in \Omega, \quad (\text{A22})$$

where $\Omega = [-0.7, 0.7]$, $\beta = 0.01$, $k = 0.7$, $u = \sin^3(6x)$, and f can be derived from (A22). For this scenario, we use 4 sensors located at $x \in \{-0.7, -0.47, 0.47, 0.7\}$ to provide noisy observations of the solution u . Additionally, we employ 40 sensors, equally spaced within $[-0.7, 0.7]$, to measure f . Both u and f measurements are assumed to contain noise. In the simulation, measurement errors are modeled as $z_i^u \sim N(0, 0.05^2)$ for $i = 1, 2, \dots, 40$, with each solution sensor providing 10 replicate measurements, and $z_i^f \sim N(0, 0.05^2)$ for $i = 1, 2, \dots, 400$.

The experimental results are summarized in Table A7. The findings exhibit a similar pattern to those in Table A5: The Bayesian and dropout methods yield inflated coverage rates and overly wide confidence intervals, whereas the EFI method achieves an accurate coverage rate and the narrowest confidence interval. For a fair comparison, we exclude cases where B-PINN converged to incorrect solutions, as these represent instances of failure in the optimization process, see Figure A6 for an instance.

A5.5 Non-linear Poisson Inverse Problem

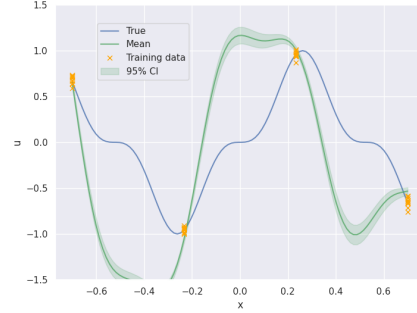
In this section, we consider the same non-linear Poisson equation as in (A22), but with $k = 0.7$ treated as an unknown parameter to be estimated. For this setup, we utilize 8 sensors, evenly distributed across $\Omega = [-0.7, 0.7]$ to measure u , with measurement noise modeled as $z_i^u \sim N(0, 0.05^2)$. Additionally, 200 sensors are employed to measure f , and these measurements are assumed to be noise-free.

Table A7: Comparison of different methods for the nonlinear 1-D Poisson model (A22) (with f -measurement error), averaged over 100 runs.

Method	hidden layers	MSE	Coverage Rate	CI-Width
PINN	[50, 50]	0.000507 (0.000044)	0.0947 (0.007564)	0.005154 (0.000565)
Dropout (0.5%)	[50, 50]	0.001050 (0.000123)	0.9962 (0.002068)	0.246367 (0.002182)
Dropout (1%)	[50, 50]	0.002807 (0.000309)	0.9861 (0.003829)	0.358088 (0.005178)
Dropout (5%)	[50, 50]	0.007010 (0.000394)	0.9956 (0.001566)	0.543565 (0.003096)
Bayesian (unstable removed)	[50, 50]	0.000376 (0.000033)	0.9938 (0.002618)	0.104673 (0.000394)
EFI	[50, 50]	0.000385 (0.000039)	0.9483 (0.009191)	0.099880 (0.002853)



(a) Success



(b) Failure

Figure A6: Successful and failed optimization results of Bayesian PINN for the nonlinear 1-D Poisson model (A22) (with f -measurement error)

To apply EFI framework to inverse problem, we extend the output of the w -network by adding an additional dimension dedicated to estimating k , as depicted in Figure A7. This modification enables the EFI framework to simultaneously estimate the solution u and the parameter k , along with their respective uncertainties. The results are presented in Table A8, demonstrating the capability of EFI to provide accurate uncertainty quantification for both u and k . In contrast, B-PINN consistently produces excessively large confidence intervals for both the solution u and the parameter k . Notably, the confidence interval for k estimated by B-PINN is approximately twice as wide as that produced by EFI, indicating a significant overestimation of uncertainty. This highlights the superior precision and robustness of the EFI framework in inverse problems.

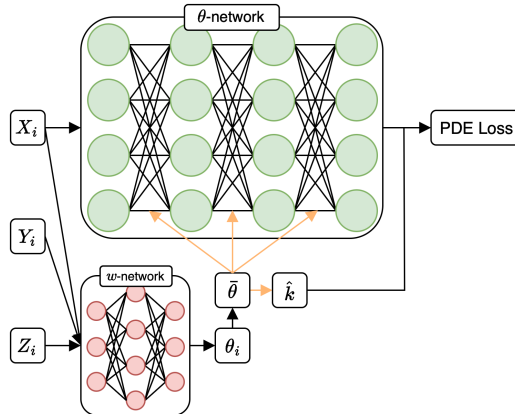


Figure A7: Diagram of EFI for inverse problems, where the orange links indicates the contribution of $\hat{\theta}$ to both θ -network and k estimation.

For this problem, we also tested EFI with a larger data modeling network, consisting of two hidden layers and each hidden layer consisting of 100 hidden units. As expected, EFI produced results

similar to those obtained with a much smaller data modeling network. As noted earlier, EFI can accurately quantify model uncertainty as long as the random errors are correctly imputed and the inverse function is consistently estimated. This capability is independent of the specific configurations of the w -network and the data modeling network, highlighting EFI’s flexibility and robustness.

Table A8: Comparison of different methods for the nonlinear 1-D Poisson model (A22) with parameter estimation, averaged over 100 runs.

Method	hidden layers	MSE($\times 10^{-4}$)	Coverage Rate	CI-Width	k-Mean	k-Coverage Rate	k-CI-Width
PINN	[50, 50]	1.79 (0.13)	0.1464 (0.0097)	0.0045 (0.0001)	0.6998 (0.0006)	0.00 (0.0000)	7.9e-5 (5e-6)
Dropout (0.5%)	[50, 50]	1.84 (0.10)	1.0000 (0.0000)	0.2393 (0.0035)	0.6912 (0.0007)	0.09 (0.0288)	0.0045 (0.0002)
Dropout (1.0%)	[50, 50]	2.72 (0.24)	1.0000 (0.0000)	0.3032 (0.0071)	0.6874 (0.0008)	0.11 (0.0314)	0.0091 (0.0007)
Dropout (5.0%)	[50, 50]	27.08 (2.12)	1.0000 (0.0000)	0.5363 (0.0131)	0.6493 (0.0023)	0.06 (0.0239)	0.0252 (0.0014)
Bayesian	[50, 50]	1.31 (0.08)	0.9752 (0.0055)	0.0517 (3e-5)	0.6994 (0.0005)	1.00 (0.0000)	0.0411 (0.0002)
EFI	[50, 50]	1.03 (0.08)	0.9473 (0.0099)	0.0396 (0.0002)	0.6985 (0.0004)	0.94 (0.0239)	0.0179 (0.0002)
EFI	[100, 100]	0.98 (0.07)	0.9560 (0.0099)	0.0395 (0.0002)	0.6995 (0.0004)	0.96 (0.0197)	0.0168 (0.0002)

A5.6 Poisson equation with unknown noise standard deviation

We now consider the case where the noise standard deviation is treated as an unknown parameter. In this setting, the variability in the observations due to the inherent randomness of the data-generating process—often referred to as systematic error—can be interpreted as aleatoric uncertainty.

As shown in Table A9, the EFI algorithm successfully recovers accurate estimates for both the solution u and the noise standard deviation σ , accompanied by well-calibrated confidence intervals.

Table A9: Numerical results for 1D-Poisson with unknown $\sigma = 0.1$ and sample size $n_b = 60$, where the number in the parentheses represents the standard error of the estimator.

Method	MSE	Coverage Rate	CI-Width	σ -mean	σ -CR	σ -CI-Width
EFI	0.000220 (0.000025)	0.9559 (0.014349)	0.061347 (0.000848)	0.097269 (0.001158)	0.9400 (0.023868)	0.056768 (0.001284)

A5.7 Black-Scholes Model

As a practical example, we consider the classical option pricing model in finance — the Black-Scholes model (Black and Scholes, 1973):

$$\begin{aligned}
\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV &= 0, \\
C(0, t) &= 0 \text{ for all } t, \\
C(S, t) &\rightarrow S - K \text{ as } S \rightarrow \infty, \\
C(S, T) &= \max\{S - K, 0\},
\end{aligned} \tag{A23}$$

which describes the price $V(S, t)$ of an option. Here, S is the price of the underlying asset (e.g., a stock), t is time, σ represents the volatility of the asset, r is the risk-free interest rate, K is the strike price, and T is the expiration time of the option. The boundary conditions reflect specific financial constraints.

This model has been widely used to calculate the price of European call and put options. Specifically, the analytic solution for the call option price $C(S_t, t)$ is given by

$$\begin{aligned}
C(S_t, t) &= \Phi(d_+)S_t - \Phi(d_-)Ke^{-r(T-t)}, \\
d_+ &= \frac{1}{\sigma\sqrt{T-t}} \left[\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t) \right], \\
d_- &= d_+ - \sigma\sqrt{T-t},
\end{aligned} \tag{A24}$$

where $\Phi(\cdot)$ denotes the standard normal cumulative distribution function. However, the uncertainty of the model has not yet been well studied in the literature. Accurately quantifying model uncertainty can significantly benefit decision-making, providing investors with a scientific foundation for making safer and more informed choices.

In this simulation experiment, we set $T = 1$, $\sigma = 0.5$, $r = 0.05$ and $K = 1$. The domain is defined on $\Omega = [0, T] \times [0, S_{\max}]$, where $S_{\max} = 2$. We assume the availability of 5 sensors at $t = 0$ for the price levels $S \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, with each sensor providing 10 replicate measurements. Measurement errors are modeled as $z_i^u \sim N(0, 0.05^2)$ for $i = 1, \dots, 50$, representing noisy observations. For the boundaries at $\{S = 0\}$ and $\{t = T\}$, we use 50 sensors with noise-free measurements. For physical domain, we randomly pick 800 points from Ω to satisfy the Black-Scholes equation. The results of the simulation are presented in Table A10, where the metrics are evaluated at $t = 0$ and $t = 0.5$. At $t = 0$, the evaluation reflects the model’s performance using noisy observed data, while at $t = 0.5$, the solutions are extended from the boundaries using the Black-Scholes equation. This setup highlights the model’s ability to handle noisy observations and accurately propagate solutions over time through the governing equation. EFI demonstrates superior performance by providing not only the most accurate solutions, as evidenced by the lowest MSE, but also the most reliable confidence intervals.

Table A10: Comparison of different methods for the Black-Scholes Model, averaged over 100 runs: ‘CR’ refers to the coverage rate with a nominal value of 95%.

Method	hidden layers	MSE($t=0$)($\times 10^{-4}$)	CR($t=0$)	CI-Width($t=0$)	MSE($t=0.5$)($\times 10^{-4}$)	CR($t=0.5$)	CI-Width($t=0.5$)
PINN	[50, 50]	3.08 (0.44)	0.1410 (0.0102)	0.0046 (0.0002)	15.73 (2.26)	0.2427 (0.0192)	0.0080 (0.0006)
Dropout (0.5%)	[50, 50]	1.37 (0.20)	0.5897 (0.0216)	0.0190 (0.0002)	2.19 (0.37)	0.6303 (0.0252)	0.0197 (0.0004)
Dropout (1.0%)	[50, 50]	4.30 (2.59)	0.6743 (0.0219)	0.0244 (0.0005)	1.83 (0.26)	0.6983 (0.0255)	0.0234 (0.0004)
Dropout (5.0%)	[50, 50]	1.71 (0.70)	0.9137 (0.0122)	0.0538 (0.0004)	1.70 (0.21)	0.9387 (0.0110)	0.0510 (0.0002)
Bayesian ($\sigma_f = 0.05$)	[50, 50]	1.59 (0.41)	0.9637 (0.0175)	0.0516 (0.0010)	12.61 (7.14)	0.9413 (0.0187)	0.0658 (0.0015)
Bayesian ($\sigma_f = 0.005$)	[50, 50]	10.75 (1.46)	0.5437 (0.0255)	0.0388 (0.0007)	39.39 (8.82)	0.4807 (0.0225)	0.0426 (0.0010)
EFI	[50, 50]	0.38 (0.05)	0.9440 (0.0133)	0.0158 (0.0001)	0.17 (0.02)	0.9600 (0.0082)	0.0123 (0.0001)

To further illustrate these findings, we visualize the prediction surface in Figure A8. The figure reveals that B-PINN extends the solution poorly toward the edge at $S = 2$, where no data points are available, relying solely on physical laws for extrapolation. In contrast, EFI provides a smoother and more accurate extension. The dropout method performs reasonably well for this example with a dropout rate of 5%; however, its confidence interval remains significantly wider than that of EFI. As previously noted, determining an appropriate dropout rate is not feasible without additional information. Figure A9 highlights EFI’s ability to correctly quantify uncertainties. Near the boundary at $S = 0$, where boundary information is available, the confidence interval is appropriately narrow. As the stock price S increases, and boundary information becomes scarce, the confidence interval widens, reflecting the growing uncertainty. In comparison, dropout and Bayesian methods fail to capture this behavior accurately. They produce overly broad or inconsistent intervals, particularly near the boundaries and regions with limited data, underscoring their limitations in handling uncertainty quantification for this problem.

A5.8 Real Data: Montroll Growth Model

Consider the Montroll growth model:

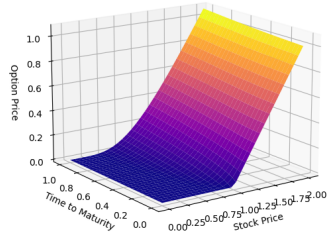
$$\frac{dp}{dt}(t) = k \cdot p(t) \cdot \left(1 - \left(\frac{p(t)}{C}\right)^\theta\right), \quad (\text{A25})$$

where k , C , and θ are unknown parameters. We applied this model to published data on the growth of Chinese hamster V79 fibroblast tumor cells (Marusic et al., 1994), which also appears in Rodrigues (2024). The dataset comprises 45 measurements of tumor volumes (10^9 vm^3) collected over a 60-day period. Table A11 shows the parameter estimates obtained using PINN, and Figure A10(a) shows the learned growth curve.

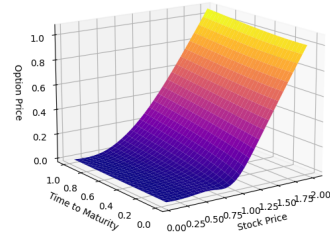
Table A11: Parameter estimates obtained with PINN for the model (A25).

Parameter	Estimated Value
k	0.8311
C	7.3327
θ	0.1694

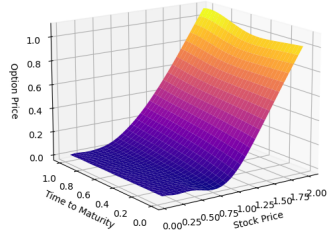
However, the standard PINN method does not provide confidence intervals for $p(t)$ or for the parameters k , C , and θ . To apply EFI-PINN to this dataset, we assume a heteroscedastic noise



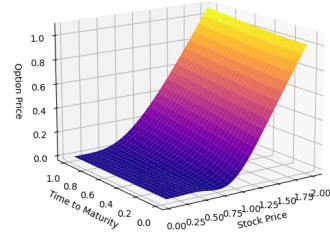
(a) True price



(b) PINN dropout (5.0%)

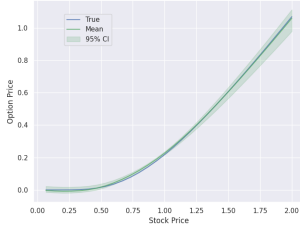


(c) Bayesian PINN ($\sigma_f = 0.05$)

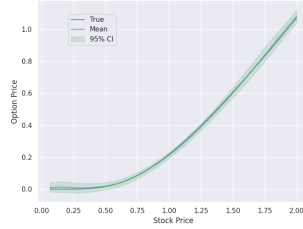


(d) EFI

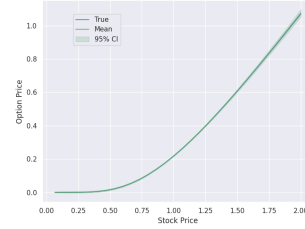
Figure A8: European Call Option Price.



(a) PINN dropout (5.0%)

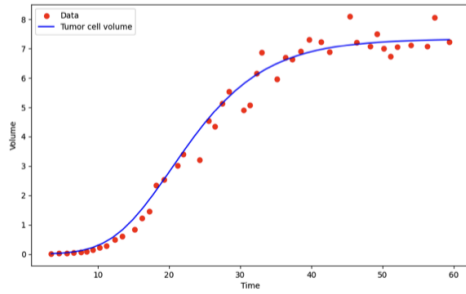


(b) Bayesian PINN ($\sigma_f = 0.05$)

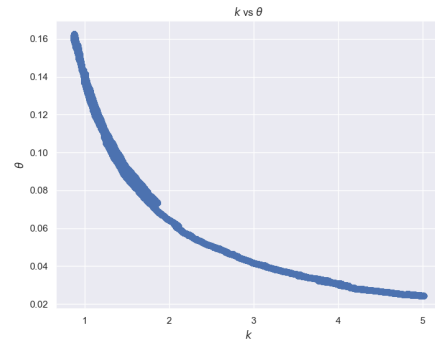


(c) EFI

Figure A9: European Call Option Price at $t = 0$.



(a) Montroll growth curve learned with PINN



(b) Relationship between k and θ

Figure A10: Montroll growth Model for Chinese hamster V79 fibroblast tumor cells.

structure given by $\sigma_t = \sqrt{t} \cdot \sigma$, where σ is treated as an additional unknown parameter. Thus, we estimate four parameters in total under the EFI framework. Using EFI-PINN, we detected that k and θ are nearly non-identifiable, see Figure A10(b), which shows their joint distribution by plotting their samples throughout training.

To address this non-identifiability issue, we fix $\theta = 0.1$ and reduce the model to:

$$\frac{dp}{dt}(t) = k \cdot p(t) \cdot \left(1 - \left(\frac{p(t)}{C}\right)^{0.1}\right).$$

The corresponding confidence intervals for $p(t)$ are shown in the left plot of Figure 3. The confidence intervals of k , C , and σ are given in Table A12.

Table A12: Parameter estimates (with 95% confidence intervals) obtained by EFI-PINN for the Montroll growth model.

Parameter	95% Confidence Interval	Mean
θ	–	0.1
k	(1.25, 1.40)	1.25
C	(7.30, 7.69)	7.44
σ	(0.27, 0.47)	0.36

The Montroll experiment highlights the strength of EFI in quantifying uncertainty for all parameters of interest. Moreover, it demonstrates EFI’s ability to detect model identifiability issues, underscoring its utility in the statistical inference of scientific models. Additionally, EFI produces an estimate of σ , which enables the quantification of predictive uncertainty.

A5.9 Real Data: FKPP Model and Porous-FKPP Model

Consider the Fisher–Kolmogorov–Petrovsky–Piskunov (FKPP) model and the porous FKPP (P-FKPP) model, which are governed by the following reaction-diffusion equations:

$$\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2} + ru \left(1 - \frac{u}{K}\right), \quad (\text{A26})$$

$$\frac{\partial u}{\partial t} = D \frac{\partial}{\partial x} \left[\left(\frac{u}{K}\right)^m \frac{\partial u}{\partial x} \right] + ru \left(1 - \frac{u}{K}\right), \quad (\text{A27})$$

where D , r , and m are unknown parameters, and K denotes the carrying capacity. This equation has been used to model a wide range of growth and transport of biological processes. We applied it to scratch assay data (Jin and Cai, 2006). The biological experiments were conducted under varying initial cell densities — specifically, 10,000, 12,000, 14,000, 16,000, 18,000, and 20,000 cells per well. Cell densities were recorded at 37 equally-spaced spatial positions across five equally-spaced time points — specifically, 0.0 days, 0.5 days, 1.0 days, 1.5 days, and 2.0 days. See also Lagergren et al. (2020) for additional descriptions of the dataset. In addition to the dataset, we partitioned the space-time domain $[0, 2] \times [0, 2]$, where the first interval corresponds to the spatial domain and the second to the temporal domain, into a 50×10 grid for computing the PDE loss (i.e., the f -term in equation (10)). The fitting curves of the EFI algorithm for different models and initial density values are shown in Figures A11 and A12. Notably, beyond point estimation, EFI also constructs confidence intervals. Additionally, we note that the right plot of Figure 3 was generated using a parameter setting different from that listed in Table A20. In this setting, fewer sample points were allocated for computing the PDE loss, which resulted in smaller fitting errors but larger deviations from the assumed PDE model. Essentially, the two settings correspond to different datasets, as the number of sample points used to evaluate the energy function differs between them.

In Table A13, the root mean squared errors (RMSEs) are computed between the predicted solution u and the observed data, reflecting a combination of epistemic and aleatoric uncertainty. Based on the model formulations in (A26) and (A27), the P-FKPP model is more flexible and is therefore expected to exhibit reduced epistemic uncertainty, leading to smaller RMSE values. Consistent with

this expectation, Table A13 shows that the P-FKPP model achieves lower RMSEs compared to the standard FKPP model.

Regarding parameter uncertainty, we note that EFI is able to quantify the uncertainty associated each parameter. However, due to the transformation applied to the first term of (A27), the values of D are no longer on the same scale across the two models, whereas the values of r remain comparable in scale. The results are reported in Table A13.

Summary Through both simulation and real data experiments, we have demonstrated that the proposed EFI algorithm effectively quantifies uncertainties associated with the model and the data-generating process, resulting in accurate estimation of both epistemic and aleatoric uncertainties.

Table A13: RMSE and estimated parameters with 95% confidence intervals for FKPP and Porous-FKPP models.

Model	Initial Cell Density	RMSE	D	Interval	R	Interval	M	Interval
FKPP	10000	58.04	0.00936	(0.00754, 0.01195)	0.829	(0.797, 0.877)	–	–
FKPP	12000	82.09	0.00378	(0.00281, 0.00461)	0.632	(0.603, 0.658)	–	–
FKPP	14000	82.93	0.02929	(0.02739, 0.03268)	0.534	(0.505, 0.585)	–	–
FKPP	16000	99.14	0.02636	(0.02503, 0.02789)	0.608	(0.585, 0.633)	–	–
FKPP	18000	115.27	0.03784	(0.03541, 0.04032)	0.549	(0.524, 0.575)	–	–
FKPP	20000	136.67	0.05471	(0.05007, 0.05817)	0.492	(0.458, 0.520)	–	–
P-FKPP	10000	46.90	1167.67	(72.48, 2719.97)	0.846	(0.832, 0.856)	1.335	(1.037, 1.490)
P-FKPP	12000	67.88	1825.54	(32.84, 4416.53)	0.674	(0.649, 0.696)	1.433	(1.033, 1.603)
P-FKPP	14000	73.59	289.37	(79.08, 552.13)	0.625	(0.600, 0.649)	1.096	(0.951, 1.199)
P-FKPP	16000	70.83	57.36	(19.21, 97.22)	0.628	(0.608, 0.650)	0.920	(0.804, 0.999)
P-FKPP	18000	96.50	21.58	(9.07, 35.78)	0.563	(0.536, 0.587)	0.780	(0.683, 0.863)
P-FKPP	20000	123.34	1.472	(1.058, 2.020)	0.496	(0.464, 0.530)	0.408	(0.370, 0.452)

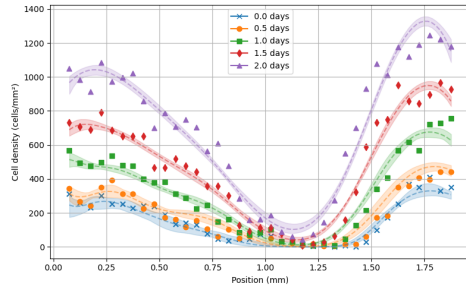
A6 Experimental Settings

In all the simulation experiments, we begin by generating data from a specific physics model. Using the simulated data, we iteratively run the algorithm to estimate the model parameters. To enhance convergence of Algorithm 1, some algorithmic parameters (such as learning rate, SGD momentum, $\lambda = 1/\epsilon$, and etc.) are adjusted during the initial iterations, referred to as the annealing period. To tune different parameters, we use three different annealing schemes, including linear, exponential and polynomial. Their specific forms are in Table A14. For the nonlinear Poisson inverse problem, both the DNN parameters ϑ and the unknown parameter k are estimated; while for all other problems, only the DNN parameters ϑ are estimated. At the end of the simulation, the samples collected in the burn-in period are discarded, and the samples collected in the remaining iterations are used for inference. The burn-in period is set to be at least as long as the annealing period across all experiments.

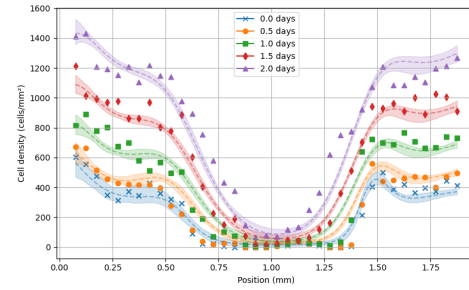
In our simulations, to ensure the weights of the w -network remain within a compact space as required in Assumption A4.1-(i), we impose a Gaussian prior, $N(0,100)$, on each connection weight. However, due to the large variance, this prior has minimal impact on the algorithm’s performance, serving primarily to ensure its stability.

Table A14: Notations

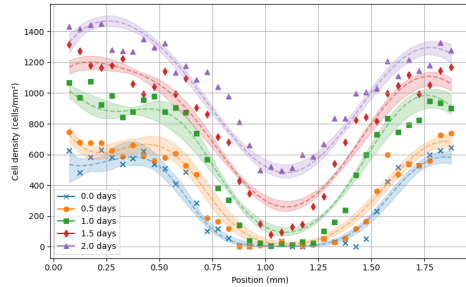
Notation	Meaning
epochs	total number of sampling/optimization iterations
burn-in period	the proportion of total iterations allocated to the burn-in process
annealing period	the proportion of total iterations allocated for parameter adjustment to enhance convergence
linear_x_y (with progress $\rho \in [0, 1]$)	$x + (y - x)\rho$
exp_x_y (with progress $\rho \in [0, 1]$)	$x^{1-\rho}y^\rho$
poly_x_c_k (with progress $\rho \in [0, 1]$)	$\frac{x}{1+(c\rho)^k}$



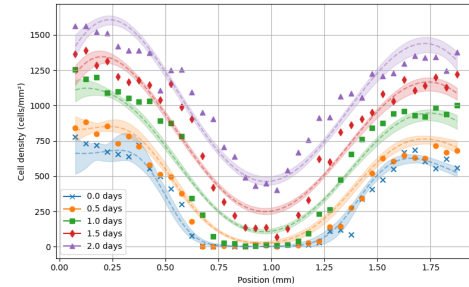
(a) Initial density 10,000



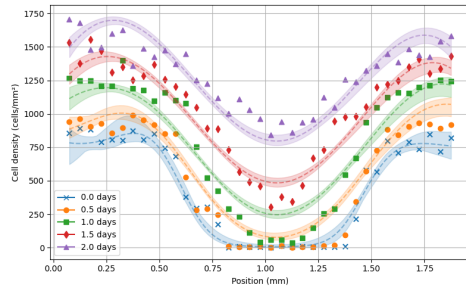
(b) Initial density 12,000



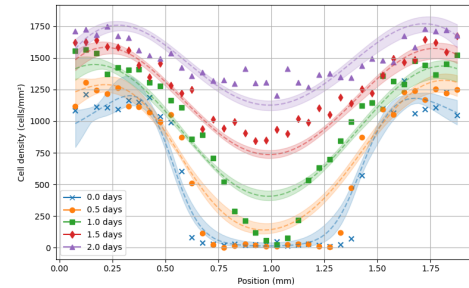
(c) Initial density 14,000



(d) Initial density 16,000



(e) Initial density 18,000

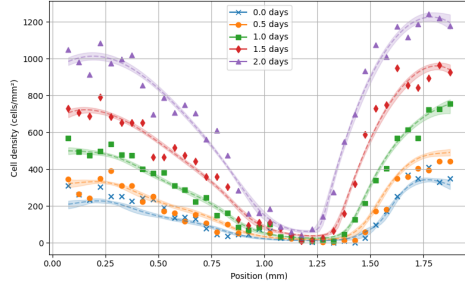


(f) Initial density 20,000

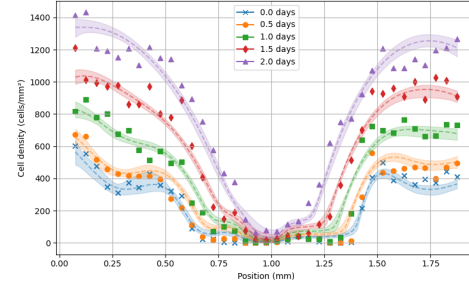
Figure A11: FKPP model

References

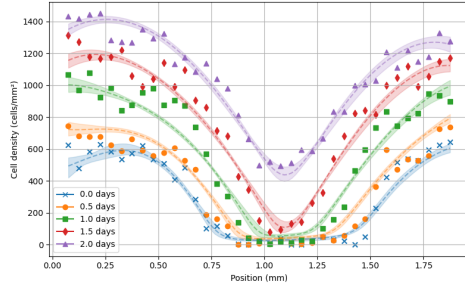
- Black, F. and Scholes, M. S. (1973), “The Pricing of Options and Corporate Liabilities,” *Journal of Political Economy*, 81, 637 – 654.
- Deng, W., Zhang, X., Liang, F., and Lin, G. (2019), “An adaptive empirical Bayesian method for sparse deep learning,” *Advances in neural information processing systems*, 32.
- Dittmer, S., King, E. J., and Maass, P. (2018), “Singular Values for ReLU Layers,” *IEEE Transactions on Neural Networks and Learning Systems*, 31, 3594–3605.
- Higham, N. J. and Cheng, S. H. (1998), “Modifying the inertia of matrices arising in optimization,” *Linear Algebra and its Applications*, 261–279.
- Jin, J. and Cai, T. T. (2006), “Estimating the Null and the Proportion of Nonnull Effects in Large-Scale Multiple Comparisons,” *Journal of the American Statistical Association*, 102, 495 – 506.
- Lagergren, J. H., Nardini, J. T., Baker, R. E., Simpson, M. J., and Flores, K. B. (2020), “Biologically-informed neural networks guide mechanistic modeling from sparse experimental data,” *PLoS Computational Biology*, 16.



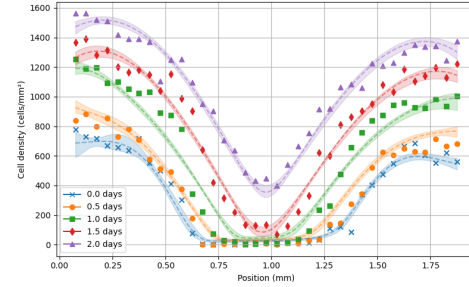
(a) Initial density 10,000



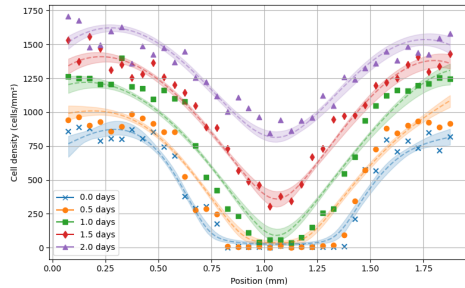
(b) Initial density 12,000



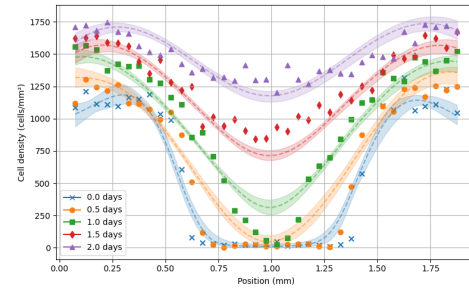
(c) Initial density 14,000



(d) Initial density 16,000



(e) Initial density 18,000



(f) Initial density 20,000

Figure A12: Porous-FKPP model

Liang, F., Kim, S., and Sun, Y. (2025), “Extended Fiducial Inference: Toward an Automated Process of Statistical Inference,” *Journal of the Royal Statistical Society Series B*, 87, 98–131.

Liang, F., Li, Q., and Zhou, L. (2018), “Bayesian Neural Networks for Selection of Drug Sensitive Genes,” *Journal of the American Statistical Association*, 113, 955–972.

Liang, S., Sun, Y., and Liang, F. (2022), “Nonlinear Sufficient Dimension Reduction with a Stochastic Neural Network,” *NeurIPS 2022*.

Marusic, M., Bajzer, Z., Freyer, J. P., and Vuk-Pavlović, S. (1994), “Analysis of growth of multicellular tumour spheroids by mathematical models,” *Cell Proliferation*, 27.

Milnor, J. and Stasheff, J. D. (1974), *Characteristic Classes*, Princeton University Press.

Rodrigues, J. A. (2024), “Using Physics-Informed Neural Networks (PINNs) for Tumor Cell Growth Modeling,” *Mathematics*, 12.

Song, Q., Sun, Y., Ye, M., and Liang, F. (2020), “Extended Stochastic Gradient MCMC for Large-Scale Bayesian Variable Selection,” *Biometrika*, 107, 997–1004.

Table A15: Parameter settings for 1D-Poisson

Parameter Name	PINN	Dropout	Bayesian PINN	EFI
t_{start}	-0.7	-0.7	-0.7	-0.7
t_{end}	0.7	0.7	0.7	0.7
noise sd in u	0.05	0.05	0.05	0.05
noise sd in f	0.0	0.0	0.0	0.0
# of solution sensors	2	2	2	2
# of solution replicates	10	10	10	10
# of differential sensors	200	200	200	200
# of differential replicates	1	1	1	1
epochs	500000	500000	100000	200000
burn-in period	0.5	0.5	0.4	0.1
hidden layers	[50, 50]	[50, 50]	[50, 50]	[50, 50]
activation function	tanh	tanh	tanh	tanh
learning rate	3e-4	3e-4	1e-4/1e-4/linear_1e-4_3e-6	poly_5e-6_100.0_0.55
η_f	1.0	1.0	/	1.0
dropout rate	/	0.5%/1%/5%	/	/
L	/	/	6	/
σ_f	/	/	0.05/exp_0.05_0.005/exp_0.05_0.0005	/
σ_u	/	/	0.05	/
annealing period	/	/	0.3	0.1
sgd momentum	/	/	/	linear_0.9_0.0
sgld learning rate	/	/	/	poly_5e-6_10.0_0.55
λ	/	/	/	linear_50.0_500.0
η_θ	/	/	/	1.0
encoder hidden layers	/	/	/	[16, 16, 16]
encoder activation function	/	/	/	leaky relu

Table A16: Parameter settings for 1D-Poisson (with f error)

Parameter Name	PINN	Dropout	Bayesian PINN	EFI
t_{start}	-0.7	-0.7	-0.7	-0.7
t_{end}	0.7	0.7	0.7	0.7
noise sd in u	0.05	0.05	0.05	0.05
noise sd in f	0.05	0.05	0.05	0.05
# of solution sensors	4	4	4	4
# of solution replicates	10	10	10	10
# of differential sensors	40	40	40	40
# of differential replicates	10	10	10	10
epochs	100000	100000	50000	200000
burn-in period	0.5	0.5	0.4	0.1
hidden layers	[50, 50]	[50, 50]	[50, 50]	[50, 50]
activation function	tanh	tanh	tanh	tanh
learning rate	3e-4	3e-4	1e-4	poly_2.5e-6_50.0_0.55
η_f	1.0	1.0	/	1.0
dropout rate	/	0.5%/1%/5%	/	/
L	/	/	6	/
σ_f	/	/	0.05	/
σ_u	/	/	0.05	/
annealing period	/	/	/	0.1
sgd momentum	/	/	/	linear_0.9_0.0
sgld learning rate	/	/	/	poly_5e-6_100.0_0.55
λ	/	/	/	linear_50.0_1000.0
η_θ	/	/	/	1.0
encoder hidden layers	/	/	/	[64, 64, 16]
encoder activation function	/	/	/	leaky relu

Sun, Y. and Liang, F. (2022), “A kernel-expanded stochastic neural network,” *Journal of the Royal Statistical Society Series B*, 84, 547–578.

Sun, Y., Song, Q., and Liang, F. (2022), “Consistent Sparse Deep Learning: Theory and Computation,” *Journal of the American Statistical Association*, 117, 1981–1995.

Table A17: Parameter settings for nonlinear 1D-Poisson (with f error)

Parameter Name	PINN	Dropout	Bayesian PINN	EFI
t_{start}	-0.7	-0.7	-0.7	-0.7
t_{end}	0.7	0.7	0.7	0.7
noise sd in u	0.05	0.05	0.05	0.05
noise sd in f	0.05	0.05	0.05	0.05
# of solution sensors	4	4	4	4
# of solution replicates	10	10	10	10
# of differential sensors	40	40	40	40
# of differential replicates	10	10	10	10
k	0.7	0.7	0.7	0.7
epochs	100000	100000	100000	200000
burn-in period	0.5	0.5	0.4	0.1
hidden layers	[50, 50]	[50, 50]	[50, 50]	[50, 50]
activation function	tanh	tanh	tanh	tanh
learning rate	3e-4	3e-4	1e-4	poly_5e-6_100.0_0.55
η_f	1.0	1.0	/	1.0
dropout rate	/	0.5%/1%/5%	/	/
L	/	/	6	/
σ_f	/	/	exp_0.2_0.05	/
σ_u	/	/	0.05	/
annealing period	/	/	0.3	0.1
sgd momentum	/	/	/	linear_0.9_0.0
sgld learning rate	/	/	/	poly_5e-6_100.0_0.55
λ	/	/	/	exp_50.0_1000.0
η_θ	/	/	/	1.0
encoder hidden layers	/	/	/	[64, 64, 16]
encoder activation function	/	/	/	leaky relu

Yang, L., Meng, X., and Karniadakis, G. E. (2021), “B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data,” *Journal of Computational Physics*, 425, 109913.

Table A18: Parameter settings for nonlinear 1D-Poisson with parameter estimation

Parameter Name	PINN	Dropout	Bayesian PINN	EFI
t_{start}	-0.7	-0.7	-0.7	-0.7
t_{end}	0.7	0.7	0.7	0.7
noise sd in u	0.05	0.05	0.05	0.05
noise sd in f	0.0	0.0	0.0	0.0
# of solution sensors	8	8	8	8
# of solution replicates	10	10	10	10
# of differential sensors	200	200	200	200
# of differential replicates	1	1	1	1
k	0.7	0.7	0.7	0.7
epochs	50000	500000	50000	350000
burn-in period	0.5	0.5	0.4	0.1
hidden layers	[50, 50]	[50, 50]	[50, 50]	[50, 50]
activation function	tanh	tanh	tanh	tanh
learning rate	3e-4	3e-4	1e-4	poly_5e-6_100.0_0.55
η_f	1.0	1.0	/	1.0
dropout rate	/	0.5%/1%/5%	/	/
L	/	/	10	/
σ_f	/	/	0.05	/
σ_u	/	/	0.05	/
annealing period	/	/	/	0.1
sgd momentum	/	/	/	linear_0.9_0.0
sgld learning rate	/	/	/	poly_5e-6_100.0_0.55
λ	/	/	/	linear_50.0_1000.0
η_θ	/	/	/	1.0
encoder hidden layers	/	/	/	[128, 128, 12]
encoder activation function	/	/	/	leaky relu

Table A19: Parameter settings for Black-Scholes Model

Parameter Name	PINN	Dropout	Bayesian PINN	EFI
S range	[0.0, 2.0]	[0.0, 2.0]	[0.0, 2.0]	[0.0, 2.0]
t range	[0.0, 1.0]	[0.0, 1.0]	[0.0, 1.0]	[0.0, 1.0]
σ	0.5	0.5	0.5	0.5
r	0.05	0.05	0.05	0.05
K	1.0	1.0	1.0	1.0
noise sd	0.05	0.05	0.05	0.05
# of price sensors	5	5	5	5
# of price replicates	10	10	10	10
# of boundary samples	50	50	50	50
# of differential samples	800	800	800	800
epochs	200000	200000	100000	300000
burn-in period	0.5	0.5	0.4	0.1
hidden layers	[50, 50]	[50, 50]	[50, 50]	[50, 50]
activation function	softplus($\beta = 5$)	softplus($\beta = 5$)	softplus($\beta = 5$)	softplus($\beta = 10$)
learning rate	3e-4	3e-4	1e-4/linear_1e-4_1e-5	poly_5e-6_100.0_0.55
η_f	1.0	1.0	/	1.0
dropout rate	/	0.5%/1%/5%	/	/
L	/	/	6	/
σ_f	/	/	0.05/exp_0.05_0.005	/
σ_u	/	/	0.05	/
pretrain epochs	/	/	5000	/
annealing period	/	/	0.3	0.1
sgd momentum	/	/	/	linear_0.9_0.0
sgld learning rate	/	/	/	poly_5e-6_100.0_0.55
sgld alpha	/	/	/	1.0
λ	/	/	/	linear_50.0_1000.0
η_θ	/	/	/	1.0
encoder hidden layers	/	/	/	[64, 64, 16]
encoder activation function	/	/	/	leaky relu

Table A20: Parameter settings for real data

Parameter Name	Montroll growth	FKPP	P-FKPP
epochs	200000	200000	200000
burn-in period	0.1	0.1	0.1
hidden layers	[50, 50]	[50, 50]	[50, 50]
activation function	softplus($\beta = 10$)	tanh	tanh
learning rate	poly_1e-6_10.0_0.55	poly_1e-6_10.0_0.55	poly_1e-6_10.0_0.55
η_f	1.0	1.0	1.0
sgd momentum	0.9	0.9	0.9
sgld learning rate	poly_1e-4_10.0_0.95	poly_1e-6_10.0_0.95	poly_1e-6_10.0_0.95
sgld alpha	1.0	1.0	1.0
λ	log_50.0_500.0	log_50.0_500.0	log_50.0_500.0
η_θ	1.0	1.0	1.0
encoder hidden layers	[32, 32, 16]	[64, 64, 16]	[64, 64, 16]
encoder activation function	leaky relu	leaky relu	leaky relu