STATISTICAL INFERENCE FOR STRUCTURALLY CONSTRAINED NONLINEAR REGRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

A fundamental challenge in using deep neural networks (DNNs) is to obtain valid prediction uncertainty quantification while retaining their flexibility. Two central goals are to construct confidence intervals for the mean regression function and prediction intervals for future observations. The former quantifies uncertainty about the underlying relationship, while the latter accounts for observational noise. Existing approaches address aspects of these goals, but valid uncertainty quantification under structural constraints remains challenging, particularly when the noise distribution is unknown. Building on Extended Fiducial Inference (EFI), we develop a framework for nonlinear regression with deep neural networks, where structural constraints are imposed to restrict the admissible function class while an unknown noise distribution is accommodated. Under the assumption that the structurally constrained function class admits a locally identifiable low-dimensional representation, we establish theoretical guarantees for inference in dense, over-parameterized neural networks without requiring sparse parameterizations. Simulation studies show near-nominal coverage for both confidence and prediction intervals across a range of constrained nonlinear regression settings, including differential equations, monotonicity, and periodicity, under Gaussian and non-Gaussian noise.

Keywords: Extended Fiducial Inference, structural constraints, nonlinear regression, flexible noise modeling, uncertainty quantification

1 INTRODUCTION

A central challenge in modern statistics is to combine expressive nonlinear modeling with principled statistical inference. Classical statistical methods provide interpretable targets, principled estimation, and uncertainty quantification, but they typically rely on restrictive model classes, prespecified likelihoods, and relatively simple noise assumptions (Wahba, 1990; Wasserman, 2006). Deep neural networks (DNNs), by contrast, can represent highly complex nonlinear relationships in high dimensions, yet they still lack a comparably general inferential framework when both the regression function and the noise mechanism are difficult to specify parametrically (Schmidt-Hieber, 2020; Bauer & Kohler, 2019). This challenge is especially pronounced in nonlinear regression under structural constraints. Formally, we observe $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ generated as

$$y_i = f^\star(\boldsymbol{x}_i) + z_i, \quad i = 1, \ldots, n, \qquad \text{subject to } \mathfrak{C}(f^\star) = 0, \tag{1}$$

where $f^\star$ is an unknown nonlinear regression function, z_i has a *possibly unknown* distribution, and $\mathfrak{C}$ encodes structural restrictions such as partial differential equations, monotonicity, periodicity, or related equality and inequality conditions. Our inferential goals are to construct confidence intervals for the mean regression function $f^\star(x)$ and prediction intervals for a future observation $Y^\star = f^\star(x) + Z^\star$, while estimating the noise distribution when it is unspecified.

Classical nonparametric regression expands beyond parametric modeling through methods such as kernel and spline smoothing, smoothing spline analysis of variance (SS-ANOVA), (generalized) additive models, and structured-sparse smoothers (Wahba, 1990; Lin & Zhang, 2006; Wahba et al., 1995; Ravikumar et al., 2009; Hastie & Tibshirani, 1986). These methods are principled and often effective in low- to moderate-complexity settings under simple error assumptions. At a high level,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

many such estimators can be expressed as solutions to penalized optimization problems of the form

$$\hat{f}_{\mathrm{NP}} = \underset{f \in \mathcal{F}}{\arg\min} \left\{ \sum_{i=1}^{n} \big(y_i - f(\boldsymbol{x}_i)\big)^2 + \lambda_{\mathrm{sm}}\, \mathcal{R}_{\mathrm{sm}}(f) \right\}, \tag{2}$$

where $\mathcal{R}_{\mathrm{sm}}$ encodes smoothness-oriented regularization and thereby restricts the admissible function class. However, standard nonparametric regression methods based on smoothness restrictions may not readily accommodate heterogeneous structural constraints. Although some classical approaches link smoothness regularization to structural restrictions through linear differential operators (Heckman & Ramsay, 2000), these formulations address only a limited subset of the constraints considered in this paper.

Notably, DNNs provide a rich and flexible class of regression functions. In structured learning settings, this flexibility can be combined with scientific knowledge through constrained neural formulations, such as physics-informed neural networks (PINNs), which encode structural restrictions using residual-based penalties (Raissi et al., 2019). A generic estimator takes the form

$$\hat{f}_{\mathrm{PINN}} = \underset{f_{\boldsymbol{\theta}} \in \mathcal{F}_{\mathrm{DNN}}}{\arg\min} \left\{ \sum_{i=1}^{n} \big(y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\big)^2 + \lambda_{\mathrm{str}}\, \mathcal{R}_{\mathrm{str}}(f_{\boldsymbol{\theta}}) \right\}, \tag{3}$$

where $\mathcal{R}_{\mathrm{str}}$ measures violations of the specified structural constraints, and $\lambda_{\mathrm{str}} > 0$ controls the strength of this *soft penalty*. However, this penalized fitting formulation does not by itself ensure calibrated uncertainty quantification. Valid statistical inference for PINNs remains challenging, particularly in the over-parameterized regime.

Existing approaches to uncertainty quantification for PINNS, or general DNNs, include Bayesian simulations (Yang et al., 2021), MC dropout (Gal & Ghahramani, 2016), and deep ensembles (Lakshminarayanan et al., 2016). However, these approaches do not necessarily produce confidence intervals with nominal frequentist coverage (Shih et al., 2025). Under exchangeability, conformal prediction (Vovk et al., 2005; Lei et al., 2018) provides finite-sample marginal coverage guarantees for future responses, but these guarantees do not directly yield confidence intervals for the underlying regression function.

Extended Fiducial Inference (EFI) quantifies parameter uncertainty by inverting data-generating equations (Liang et al., 2025; Liang, 2026). In its neural implementation, the inverse map, $\boldsymbol{\theta} = G_{\boldsymbol{w}}(\mathcal{D}, \boldsymbol{Z}_n)$, is approximated by a neural network with weights $\boldsymbol{w}$, where $\boldsymbol{Z}_n$ collects the unobserved random errors in the data-generating equations. The inverse map is learned jointly with the imputation of these latent errors. For additive-noise regression under structural constraints, consider the energy function

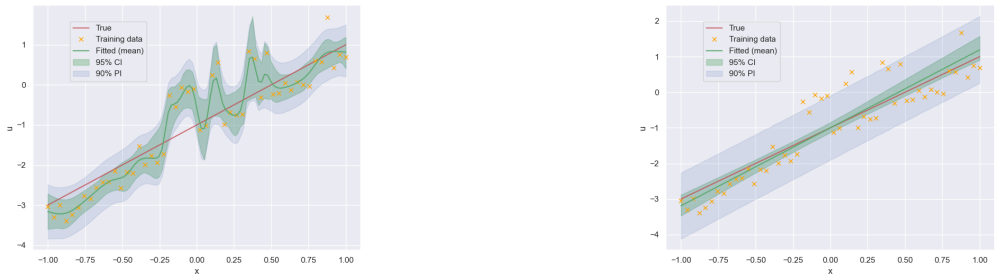
$$U(\boldsymbol{\theta}; \mathcal{D}, \boldsymbol{Z}_n) = \sum_{i=1}^{n} \big(y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - z_i\big)^2 + \mathcal{R}_{\mathrm{str}}(f_{\boldsymbol{\theta}}), \qquad z_i \overset{\text{i.i.d.}}{\sim} \pi_0, \tag{4}$$

where π_0 is the error distribution and $\mathcal{R}_{\mathrm{str}}$ is a nonnegative measure of structural constraint violations. EFI alternates between updating the inverse-network weights $\boldsymbol{w}$ and simulating the latent errors from the density

$$p_n^{\epsilon}(\boldsymbol{z} \mid \mathcal{D}, \boldsymbol{w}) \propto \pi_0^{\otimes n}(\boldsymbol{z}) \exp\Big\{ -U\big(G_{\boldsymbol{w}}(\mathcal{D}, \boldsymbol{z}); \mathcal{D}, \boldsymbol{z}\big)/\epsilon \Big\},$$

where $\pi_0^{\otimes n}(\boldsymbol{z}) = \prod_{i=1}^{n} \pi_0(z_i)$ and $\epsilon > 0$ is a small temperature parameter. The energy term favors pairs of parameter values and latent errors that satisfy the data equations and structural constraints, especially when $\epsilon \to 0$, i.e., $U(G_{\boldsymbol{w}}(\mathcal{D}, \boldsymbol{z}); \mathcal{D}, \boldsymbol{z}) \to 0$. As the latent errors are simulated, the inverse network generates a corresponding sequence of parameter values through the learned inverse map $G_{\boldsymbol{w}}(\cdot)$. Samples retained after convergence provide the basis for fiducial point estimation and uncertainty quantification. EFI thus propagates uncertainty in the latent observational errors, while accounting for the observed data and structural constraints, to the model parameters.

This paper aims to provide a general inferential framework for nonlinear regression under heterogeneous structural constraints. Building on prior EFI work for PINNs and nonparametric-error inference (Shih et al., 2025; Wang & Liang, 2026), we develop a framework for nonlinear regression using dense neural networks, incorporating structural constraints and allowing an unknown noise distribution. Figure 1 illustrates the role of structural information: imposing $f''(x) = 0$ excludes nonlinear fits that track observational noise and confines inference to the admissible class of linear regression functions.



(a) Without structural constraint

(b) With structural constraint

Figure 1: Motivating illustration of structural class reduction. The constraint $f''(x) = 0$ restricts the admissible class to linear functions, preventing the fit from absorbing local noise patterns. Detailed experimental settings are given in Appendix F.1.

More generally, the framework supports nonparametric inference under structural constraints and uncertainty quantification for dense, over-parameterized neural networks. Its theoretical justification relies on a locally identifiable low-dimensional representation of the constrained function class, together with the regularity conditions required for EFI.

We make three main contributions. First, we develop a unified EFI framework for structurally constrained nonlinear regression with an unspecified noise distribution, providing both confidence intervals for the mean regression function and prediction intervals for future observations.

Second, assuming that the constrained function class admits a locally identifiable low-dimensional representation, we establish theoretical guarantees for transferring uncertainty statements from the reduced model to identifiable functionals of the original dense neural network, without requiring a sparse parameterization.

Third, we evaluate the framework through numerical experiments involving differential and non-differential constraints under Gaussian and non-Gaussian noise, demonstrating near-nominal coverage for both confidence and prediction intervals.

2 Background

2.1 Restricted Function Classes, Spline Constructions, and Penalized Estimators

A classical approach to structurally constrained nonparametric regression uses restricted function classes, often represented through splines or related basis expansions (Hastie et al., 2009). For sufficiently simple restrictions, basis functions can be constructed to satisfy the constraints directly, reducing estimation to a finite-dimensional problem within the admissible class. Suitable spline and related bases can, for example, accommodate prescribed smoothness and certain differential restrictions (Heckman & Ramsay, 2000).

For more complex restrictions, direct basis construction can be difficult. A common alternative is to impose structure through penalties defined on a suitable function space. Classical smoothing spline estimators, for example, take the form of (2), with a Sobolev-type roughness penalty controlling smoothness (Wahba, 1990; Green & Silverman, 1994). Although effective for smoothness regularization, extending these formulations to heterogeneous structural constraints requires appropriate penalty constructions and tuning. The resulting fitting criterion does not, by itself, establish valid uncertainty quantification for general constrained models.

2.2 PINNs, Bayesian PINNs, and Related Constrained Neural Formulations

DNNs provide flexible function representations for complex constrained regression problems. PINNs incorporate structural information through residual penalties associated with governing equations,

boundary conditions, or related constraints (Raissi et al., 2019). The generic formulation in (3) balances agreement with the observed data against violations of the specified structural restrictions.

Bayesian PINNs place priors on network parameters and specify probabilistic models for observations and structural residuals (Yang et al., 2021). A schematic posterior takes the form

$$\pi(\boldsymbol{\theta} \mid \mathcal{D}) \propto \exp\{-\mathcal{L}_{\text{data}}(f_{\boldsymbol{\theta}}) - \lambda_{\text{str}}\mathcal{R}_{\text{str}}(f_{\boldsymbol{\theta}})\}\, \pi(\boldsymbol{\theta}), \tag{5}$$

where $\mathcal{L}_{\text{data}}$ represents the negative log-likelihood contribution from the observations, $\lambda_{\text{str}}\mathcal{R}_{\text{str}}$ represents the corresponding contribution from structural residuals, and $\pi(\boldsymbol{\theta})$ is the parameter prior. These formulations provide a Bayesian treatment of uncertainty, with posterior inference depending on the specified likelihoods, priors, and treatment of structural information.

2.3 Extended Fiducial Inference and Prior EFI Developments

EFI quantifies parameter uncertainty by inverting data-generating equations and propagating latent-error uncertainty through the inverse map (Liang et al., 2025), as described in Section 1. Relevant developments include EFI for PDE-informed neural models (Shih et al., 2025) and EFI with non-parametric error modeling in low-dimensional models (Wang & Liang, 2026). Building on these developments, the present paper studies dense nonlinear regression under general structural constraints with an unknown noise distribution. Appendix C provides further background on EFI and these extensions.

3 A Unified EFI Framework for Structurally Constrained Nonlinear Regression

Suppose we observe $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. We consider nonlinear regression with unknown target function $f^\star$, and assume that $f^\star$ belongs to a constrained function class determined by a collection of structural restrictions. These restrictions may include PDEs, inequality constraints, monotonicity, periodicity, and equality conditions. To connect the bundled operator $\mathfrak{C}$ in Eq. (1) with computation, we represent it through a collection of constraint families evaluated at possibly different sets of points. In particular, $\mathfrak{C}(f^\star) = 0$ means that $C^{(\ell)}(f^\star; \tilde{\boldsymbol{x}}_j^{(\ell)}) = 0$ for all equality families $\ell \in \mathcal{E}$ and all associated evaluation points, while $C^{(\ell)}(f^\star; \tilde{\boldsymbol{x}}_j^{(\ell)}) \le 0$ for all inequality families $\ell \in \mathcal{I}$ and their associated evaluation points.

Let the family index set partition as $\{1, \dots, L\} = \mathcal{E} \cup \mathcal{I}$, with $\mathcal{E}$ collecting the equality families and $\mathcal{I}$ the inequality families, and let $\{\tilde{\boldsymbol{x}}_j^{(\ell)}\}_{j=1}^{m_\ell}$ denote the evaluation points associated with the ℓth family. The structural equations are

$$\begin{aligned} y_i &= f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + z_i, && i = 1, \dots, n, \\ 0 &= C^{(\ell)}\left(f_{\boldsymbol{\theta}}; \tilde{\boldsymbol{x}}_j^{(\ell)}\right), && j = 1, \dots, m_\ell,\ \ell \in \mathcal{E}, \\ 0 &\ge C^{(\ell)}\left(f_{\boldsymbol{\theta}}; \tilde{\boldsymbol{x}}_j^{(\ell)}\right), && j = 1, \dots, m_\ell,\ \ell \in \mathcal{I}, \end{aligned} \tag{6}$$

where $\boldsymbol{\theta}$ denotes the dense network parameter, the z_i are i.i.d. latent noise variables from π_0, and each $C^{(\ell)}$ specifies one family of structural restrictions. Here π_0 denotes the reference noise law; it need not be Gaussian and may be unknown.

This formulation allows multiple types of structural information to be handled within a single framework. For example, one may take

$$C^{(\ell)}(f; \tilde{\boldsymbol{x}}) = \begin{cases} \mathcal{L}f(\tilde{\boldsymbol{x}}) - g(\tilde{\boldsymbol{x}}), & \text{PDE equality constraint,} \\ f(\tilde{\boldsymbol{x}} + P) - f(\tilde{\boldsymbol{x}}), & \text{periodicity equality constraint,} \\ -f'(\tilde{\boldsymbol{x}}), & \text{monotonicity inequality constraint.} \end{cases}$$

The next step is to translate these structural equations into an EFI energy. The guiding principle is to keep observational noise and structural restrictions distinct: latent noise enters only through the observational equation, while the structural conditions remain deterministic. Under the EFI

framework, the dense network parameter $\boldsymbol{\theta}$ in the introductory energy (4) is realized through a learned inverse map $G_{\boldsymbol{w}}$ with weights w (see the hypernetwork architecture in Figure C.2, Appendix C). For each triplet $(\boldsymbol{x}_i, y_i, z_i)$, this inverse map produces a per-sample dense parameter

$$\hat{\boldsymbol{\theta}}_i = G_{\boldsymbol{w}}(\boldsymbol{x}_i, y_i, z_i), \qquad \bar{\boldsymbol{\theta}}_n = \frac{1}{n}\sum_{i=1}^{n} \hat{\boldsymbol{\theta}}_i.$$

Thus, uncertainty in $\boldsymbol{Z}_n$ induces uncertainty in $\boldsymbol{\theta}$ through the learned inverse map, while computation proceeds through joint learning of $\boldsymbol{w}$ and sampling of $\boldsymbol{Z}_n$.

To incorporate the structural restrictions into this construction, we define a constraint-violation function $V^{(\ell)} : \mathbb{R} \to \mathbb{R}_{\geq 0}$ for each family,

$$V^{(\ell)}(c) = \begin{cases} c^2, & \ell \in \mathcal{E}, \\ \mathrm{ReLU}(c)^2, & \ell \in \mathcal{I}, \end{cases}$$

which equals zero exactly when the ℓ-th constraint is satisfied at the evaluation point. In this notation, the structural term in the introductory EFI criterion is written explicitly as a sum of compositions $V^{(\ell)} \circ C^{(\ell)}$ over constraint families and evaluation points. We then write the EFI energy as

$$U(\boldsymbol{w}; \mathcal{D}, \boldsymbol{Z}_n) = \underbrace{\sum_{i=1}^{n} \left(y_i - f_{\bar{\boldsymbol{\theta}}_n}(\boldsymbol{x}_i) - z_i\right)^2}_{\text{data}} + \eta_{\text{str}} \underbrace{\sum_{\ell=1}^{L} \sum_{j=1}^{m_\ell} V^{(\ell)}\left(C^{(\ell)}(f_{\bar{\boldsymbol{\theta}}_n}; \tilde{\boldsymbol{x}}_j^{(\ell)})\right)}_{\text{constraints}} + \eta_\theta \underbrace{\sum_{i=1}^{n} \|\hat{\boldsymbol{\theta}}_i - \bar{\boldsymbol{\theta}}_n\|^2}_{\text{consistency}}. \quad (7)$$

The first term accounts for observational noise, the second enforces the structural restrictions, and the third encourages agreement among the local inverse outputs. This decomposition keeps the data-driven uncertainty contribution distinct from the structural contribution, while the terms remain coupled through the inverse map: the latent variables z_i enter explicitly through the data equation and implicitly affect the constraint term through the induced parameter $\bar{\boldsymbol{\theta}}_n$. The positive constants η_{str} and η_θ are computational scaling parameters used to stabilize optimization and sampling; in practice, they may be annealed from smaller to larger values. They are *not* penalty-balancing parameters in the sense of PINN-type estimators in (3). Since all three terms in (7) are nonnegative, positive rescaling does not change the zero-energy feasibility set determined by the data equation, structural constraints, and consistency condition. Thus, η_{str} and η_θ affect only the numerical dynamics, not the structural constraints that define the inferential target. Appendix E.1 examines the finite-temperature sensitivity to η_θ, including its effects on consistency residuals, interval widths, and coverage.

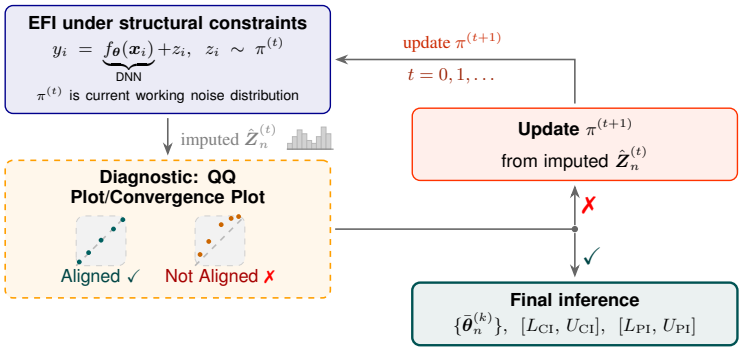


Figure 2: Workflow of the proposed EFI framework for structurally constrained nonlinear regression. The working noise distribution $\pi^{(t)}$ is fixed at π_0 when the reference law is known and may be refined from imputed latent variables when it is unknown. In $t > 1$, convergence is checked by a convergence plot of imputed latent z_i's from t and $t-1$; see, e.g., Appendix C.4 and Figure D.4 (d).

The workflow of the proposed method is illustrated in Figure 2 and summarized in Algorithm 1. We write $\pi^{(t)}$ for the working noise distribution used in refinement run t. When the reference law π_0 is known, $\pi^{(t)} = \pi_0$ for all runs. When the reference law is unknown, $\pi^{(t)}$ may be updated from the imputed latent samples, for example by Kernel Density Estimators (KDE) as in the Nonparametric-Error EFI (NE-EFI) implementation reviewed in Appendix C.3. Within each run, the latent variables

$\boldsymbol{Z}_n$ and the inverse-map weights $\boldsymbol{w}$ are updated alternately under the EFI energy $U(\boldsymbol{w}; \mathcal{D}, \boldsymbol{Z}_n)$. After the final run, the resulting samples of $\bar{\boldsymbol{\theta}}_n$ are collected as fiducial samples and used for point estimation and interval construction. A suite of self-diagnostic tools are provided in Appendix C.4

Algorithm 1: EFI Computation for Structurally Constrained Nonlinear Regression

1. **Initialize the working latent noise distribution.** Set $t = 0$. If the reference law π_0 is known, set $\pi^{(0)} = \pi_0$. Otherwise, initialize $\pi^{(0)}$ with a simple working choice (e.g. $\mathcal{N}(0, \sigma^2)$).
2. **Run one EFI stage under the current $\pi^{(t)}$.** For the current working distribution $\pi^{(t)}$, alternate the following two updates until convergence:
   (a) **Latent noise imputation.** Given the current inverse-map weights $\boldsymbol{w}$, obtain imputed samples $\hat{\boldsymbol{Z}}_n^{(t)}$ by Stochastic gradient Langevin dynamics (SGLD) targeting
   $$p^{\epsilon}(\boldsymbol{Z}_n \mid \mathcal{D}, \boldsymbol{w}) \propto \{\pi^{(t)}\}^{\otimes n}(\boldsymbol{Z}_n) \exp\{-U(w; \mathcal{D}, \boldsymbol{Z}_n)/\epsilon\},$$
   where $\epsilon > 0$ is a small temperature parameter.
   (b) **Inverse-map optimization.** Given the current imputed samples $\hat{\boldsymbol{Z}}_n^{(t)}$, update the inverse-map weights $\boldsymbol{w}$ by stochastic gradient descent (SGD) using the gradient of $U(\boldsymbol{w}; \mathcal{D}, \hat{\boldsymbol{Z}}_n^{(t)})/\epsilon$ with respect to $\boldsymbol{w}$.
3. **Check the adequacy of $\pi^{(t)}$.** Compare the imputed samples $\hat{\boldsymbol{Z}}_n^{(t)}$ with the current working distribution $\pi^{(t)}$, for example using QQ plots to check adequacy of $\pi^{(0)}$ or convergence plots $\hat{\boldsymbol{Z}}_n^{(t)}$ v.s. $\hat{\boldsymbol{Z}}_n^{(t-1)}$ to check convergence in $t > 1$.
4. **Refine $\pi^{(t)}$ if needed.** If the diagnostic indicates inadequacy, estimate a refined working distribution $\pi^{(t+1)}$ from $\hat{\boldsymbol{Z}}_n^{(t)}$ using KDE or another suitable distributional update, set $t \leftarrow t + 1$, and repeat Steps 2–4. Otherwise, proceed to inference.
5. **Construct fiducial inference.** After the final run converges, collect the resulting samples of $\bar{\boldsymbol{\theta}}_n$ as fiducial samples and use them for point estimation and interval construction.

4 Theoretical Justification

The preceding section formulates structurally constrained nonlinear regression as an EFI problem in which the structural restrictions enter the feasibility relation, thereby restricting the admissible function class rather than merely acting as a soft penalty. We next formalize the main theoretical implication of this construction: we study the regime in which the constrained function class induced by the structural restriction admits a locally identifiable low-dimensional representation, even though the ambient neural-network parameter remains high-dimensional and may be nonidentifiable. Under this reduction, EFI for the dense constrained neural model is equivalent to EFI for the corresponding low-dimensional constrained model, so uncertainty statements from low-dimensional EFI theory transfer to identifiable functionals of the regression function.

Assumption 4.1 (Locally identifiable low-dimensional constrained class)**.** Let $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}$ be a neural network parameterized by $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^p$, where p may be much larger than n, and let $\mathfrak{C}$ denote the structural constraint operator in (1). Define

$$\Theta_c = \{\boldsymbol{\theta} \in \Theta : \mathfrak{C}(f_{\boldsymbol{\theta}}) = 0\}, \qquad \mathcal{F}_c = \{f_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta_c\},$$

and assume that there exist $q < n$, a set $\Phi \subseteq \mathbb{R}^q$, and measurable maps $\rho : \Theta_c \to \Phi$ and $g : \mathcal{X} \times \Phi \to \mathbb{R}$ such that, for every $\boldsymbol{\theta} \in \Theta_c$,

$$f_{\boldsymbol{\theta}}(\cdot) = g(\cdot, \rho(\boldsymbol{\theta})). \tag{8}$$

Moreover, assume that the effective representation is locally identifiable: for every $\boldsymbol{\varphi}_0 \in \rho(\Theta_c)$, there exists an open neighborhood $V_{\boldsymbol{\varphi}_0} \subseteq \mathbb{R}^q$ of $\boldsymbol{\varphi}_0$ such that, for all $\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2 \in V_{\boldsymbol{\varphi}_0} \cap \rho(\Theta_c)$,

$$g(\cdot, \boldsymbol{\varphi}_1) = g(\cdot, \boldsymbol{\varphi}_2) \quad \Longrightarrow \quad \boldsymbol{\varphi}_1 = \boldsymbol{\varphi}_2. \tag{9}$$

Remark 4.2 (Interpretation of Assumption 4.1). Assumption 4.1 can be satisfied by constructing a common low-dimensional affine family for the neural-network parameters that represents the

constrained function class. Specifically, suppose that there exist fixed $\mathbf{a}_n \in \mathbb{R}^p$ and $\Xi_n \in \mathbb{R}^{q\times p}$, with $q < n$, such that every $f_{\boldsymbol{\theta}} \in \mathcal{F}_c$ has a representation

$$f_{\boldsymbol{\theta}}(\cdot) = f_{\mathbf{a}_n + \Xi_n^\top \rho(\boldsymbol{\theta})}(\cdot), \qquad \rho(\boldsymbol{\theta}) \in \Phi \subseteq \mathbb{R}^q,$$

where the coordinate map ρ is measurable and the affine parameter vector belongs to Θ_c. Defining $g(x, \boldsymbol{\varphi}) := f_{\mathbf{a}_n + \Xi_n^\top \boldsymbol{\varphi}}(x)$ then verifies (8). The local identifiability condition (9) can be verified by a full-rank Jacobian of function evaluations, as in Lemma B.1.

In EFI terms, one seeks a common low-dimensional affine parameter family within which the data-generating equations and structural constraints admit zero-energy solutions. This affine representation can be implemented using the narrow-neck $\boldsymbol{w}$-network developed by Shih et al. (2025); see Figure C.2 and Appendix B. The ambient parameter can therefore remain high-dimensional, while the admissible functions and zero-energy solutions are represented through low-dimensional coordinates. For smooth local representations, retaining q independent directions requires $d_h \geq q$, where d_h represents the width of the narrow-neck layer; see (B.1). Since q is generally unknown, in practice we can select a relatively small narrow-neck width d_h for which a zero-energy solution can be achieved, satisfying both the data-generating equations and the structural constraints. A larger d_h is also permissible. The ablation results in Table E.6 provide empirical evidence of robustness to the choice of d_h over the tested range.

Theorem 4.3 (EFI reduction for dense constrained neural networks)**.** *Let Assumption 4.1 hold, and let $\Phi_c = \rho(\Theta_c) \subseteq \Phi$. Then, for any $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and $\boldsymbol{Z}_n = (z_1, \ldots, z_n)$, the dense constrained model*

$$y_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + z_i, \qquad \boldsymbol{\theta} \in \Theta_c,$$

and the reduced model

$$y_i = g(\boldsymbol{x}_i, \boldsymbol{\varphi}) + z_i, \qquad \boldsymbol{\varphi} \in \Phi_c, \tag{10}$$

satisfy the same feasibility condition: there exists $\boldsymbol{\theta} \in \Theta_c$ satisfying the dense equations if and only if there exists $\boldsymbol{\varphi} \in \Phi_c$ satisfying the reduced equations. Thus, the dense and reduced formulations determine the same latent compatibility set.

Corollary 4.4 (Uncertainty quantification for quantities of interest)**.** *Under the conditions of Theorem 4.3, let $\boldsymbol{\varphi}^* = \rho(\boldsymbol{\theta}^*)$ denote the effective representation of the true constrained regression function, and let $b : \Phi_c \to \mathbb{R}$ be any quantity of interest. If EFI applies to the reduced model (10) for inference on $b(\boldsymbol{\varphi}^*)$, then the corresponding EFI uncertainty statement transfers to the dense constrained model for $b(\rho(\boldsymbol{\theta}^*))$.*

5 Simulation Studies

We evaluate the EFI-based framework for structurally constrained nonlinear regression through differential, periodic, and higher-dimensional examples under Gaussian and non-Gaussian noise. Over 100 independent replications, we report Mean Squared Error (MSE) for $f^\star$, coverage and width of 95% confidence intervals (CIs) for $f^\star(x)$, and coverage and width of 90% prediction intervals (PIs) for future responses Y^*. Interval widths are compared in conjunction with coverage relative to the nominal levels. Full results, sensitivity analyses, and implementation details are provided in Appendices D, E, F, and G.

5.1 Differential Constraints: 1-D Poisson Equation with Bimodal Mixture Gaussian Errors

We consider a 1-D Poisson equation used in prior constrained neural inference studies (Yang et al., 2021; Shih et al., 2025), with true solution $f^\star(x) = \sin^3(6x)$ on $[-0.7, 0.7]$ and deterministic constraint $0.01 f''(x) = g(x)$ enforced at $m = 200$ collocation points. Observations follow $y_i = f^\star(x_i) + z_i$, $n = 100$, with bimodal mixture noise $0.5\mathcal{N}(-0.6, 0.2^2) + 0.5\mathcal{N}(0.6, 0.2^2)$. The bimodal mixture is used only to generate the data; its distributional form is not supplied to any method. We therefore compare with standard BPINNs only under Gaussian noise, where their Gaussian observation model is correctly specified (Appendix D.2). We focus on constrained neural baselines because classical smoothers are not naturally suited to incorporating general differential operators.

Table 1 shows that dropout uncertainty is sensitive to the penalty weight and dropout rate, while EFI remains stable across the reported η_{str} settings since its value only affects the numerical dynamics. Deep ensembles severely under-cover in CI despite competitive point estimation, and subagging only partially improves CI coverage. Their Gaussian-working-model PIs over-cover and are wider than EFI's, with or without subagging. SCP attains near-nominal PI coverage but also yields wider intervals. EFI achieves near-nominal CI and PI coverage with the narrowest intervals among methods attaining comparable coverage. Appendix Table D.1 reports additional settings; the Gaussian comparison in Appendix D.2 shows BPINN's sensitivity to the differential-residual scale.

Table 1: 1-D Poisson with bimodal mixture noise. Mean (SE) over 100 replications.

Method	MSE ↓	CI Cov. (95%)	CI Width ↓	PI Cov. (90%)	PI Width ↓
EFI (η_{str}=1)	0.0045 (0.0006)	0.9545 (0.0153)	0.2925 (0.0035)	0.8957 (0.0040)	1.7269 (0.0059)
EFI (η_{str}=1 → 10 at 10%)	0.0043 (0.0006)	0.9463 (0.0166)	0.2791 (0.0033)	0.8951 (0.0038)	1.7232 (0.0059)
PINN (λ_{str}=1, dropout 1%)	0.0199 (0.0011)	0.6688 (0.0180)	0.2690 (0.0041)	0.9388 (0.0032)	1.9870 (0.0069)
PINN (λ_{str}=1, dropout 5%)	0.0197 (0.0011)	0.9278 (0.0082)	0.5193 (0.0053)	0.9467 (0.0031)	2.0255 (0.0068)
PINN (λ_{str}=10, dropout 1%)	0.0126 (0.0009)	0.8061 (0.0170)	0.2855 (0.0015)	0.9568 (0.0026)	2.0221 (0.0066)
PINN (λ_{str}=10, dropout 5%)	0.0129 (0.0009)	0.9884 (0.0033)	0.6021 (0.0034)	0.9653 (0.0023)	2.0708 (0.0063)
PINN (λ_{str}=100) + SCP	0.0205 (0.0017)	–	–	0.9027 (0.0048)	1.8527 (0.0143)
Ensemble (λ_{str}=1000, M=50)	0.0084 (0.0009)	0.0038 (0.0006)	0.0008 (0.0000)	0.9681 (0.0023)	2.0403 (0.0064)
Ensemble + Subagging (λ_{str}=100, M=50)	0.0092 (0.0009)	0.6279 (0.0286)	0.1662 (0.0013)	0.9650 (0.0025)	2.0300 (0.0064)

5.2 Non-Differential Constraints: Periodic Function

The second example considers a non-smooth periodic regression, $f^\star(x) = |\sin(2\pi x)| + 0.3\sin(4\pi x)$ on $[-0.5, 0.5]$, with equality constraint $f(\tilde{x}) - f(\tilde{x} + P) = 0$, $P = 0.5$. This example shows that the framework is not restricted to differential constraints. Appendix F.3 reports a monotonicity example, corresponding to the inequality-type restrictions discussed earlier.

Table 2 shows that classical smoothers attain near-nominal coverage, although their fitted functions are not constrained to satisfy periodicity. Under Gaussian noise, dropout, BPINN, and deep ensembles under-cover in CI, while their PI coverage remains close to the nominal level. Under bimodal noise, dropout exhibits substantial CI undercoverage and PI overcoverage, with wider PIs than EFI. SCP attains nominal PI coverage for Y^*, but it does not address calibrated inference for $f^\star$ and is less efficient under the non-Gaussian regime. EFI attains near-nominal CI and PI coverage in both Gaussian and bimodal regimes while retaining substantially narrower valid intervals under non-Gaussian noise.

Table 2: Periodic model under Gaussian and bimodal mixture noise. Mean (SE) over 100 replications.

Method	MSE ↓	CI Cov. (95%)	CI Width ↓	PI Cov. (90%)	PI Width ↓
Gaussian noise					
EFI	0.0005 (0.0000)	0.9412 (0.0006)	0.1221 (0.0007)	0.8926 (0.0004)	0.3492 (0.0007)
Cubic Smoothing Spline	0.0010 (0.0001)	0.9450 (0.0044)	0.1141 (0.0033)	0.8973 (0.0038)	0.3519 (0.0028)
Cubic Local Polynomial	0.0013 (0.0000)	0.9478 (0.0033)	0.1322 (0.0006)	0.8955 (0.0039)	0.3757 (0.0031)
PINN (λ_{str}=1, dropout 5%)	0.0007 (0.0000)	0.9261 (0.0054)	0.1224 (0.0009)	0.8870 (0.0039)	0.3313 (0.0012)
PINN (λ_{str}=1) + SCP	0.0046 (0.0004)	–	–	0.9007 (0.0040)	0.3951 (0.0040)
BPINN (σ_{diff}=1)	0.0010 (0.0000)	0.7834 (0.0095)	0.0683 (0.0004)	0.8865 (0.0035)	0.3335 (0.0001)
Ensemble (λ_{str}=100, M=50)	0.0006 (0.0000)	0.6443 (0.0111)	0.0575 (0.0025)	0.8965 (0.0038)	0.3395 (0.0013)
Bimodal mixture noise					
EFI	0.0041 (0.0003)	0.9402 (0.0100)	0.2972 (0.0070)	0.9103 (0.0030)	1.7617 (0.0041)
Cubic Smoothing Spline	0.0347 (0.0038)	0.9468 (0.0055)	0.6596 (0.0257)	0.8921 (0.0042)	1.8829 (0.0132)
Cubic Local Polynomial	0.0505 (0.0015)	0.9441 (0.0039)	0.8333 (0.0031)	0.8904 (0.0039)	2.2067 (0.0144)
PINN (λ_{str}=1, dropout 5%)	0.0185 (0.0007)	0.6275 (0.0111)	0.2224 (0.0021)	0.9351 (0.0032)	1.9552 (0.0047)
PINN (λ_{str}=10) + SCP	0.0297 (0.0017)	–	–	0.8963 (0.0038)	1.8666 (0.0111)

5.3 High-Dimensional Structurally Constrained Regression with A Three-Way Interaction

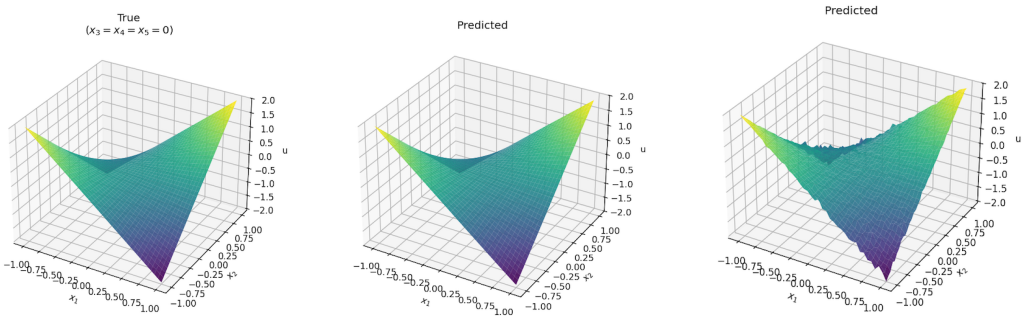
The third example tests high-dimensional nonlinear structure rather than noise misspecification. We consider $f^\star(\boldsymbol{x}) = 2x_1x_2 + 3x_3^2 + x_1x_2x_4 + 2x_5$ on $[-1, 1]^5$ with Gaussian noise and equality

constraints on selected higher-order derivatives. The three-way interaction $x_1x_2x_4$ makes the problem challenging for classical smoothers such as SS-ANOVA, whose default restrictions do not directly target this interaction structure. BPINNs are omitted from this comparison because the implementation did not train reliably in this higher-dimensional constrained setting.

Table 3 shows that SS-ANOVA under-covers for $f^\star$ despite Gaussian noise, while PINN dropout over-covers and Deep Ensembles under-cover. EFI attains near-nominal CI coverage with competitive MSE, estimates the noise variance σ^2 jointly with the DNN parameter $\boldsymbol{\theta}$, and remains close to nominal levels, with slight CI overcoverage and PI undercoverage. The learned surfaces in Figure 3 show this difference visually: EFI recovers the main interaction patterns with smoother surfaces, whereas the PINN fit exhibits more local variation around the noisy observations. We attribute this smooth generalization to EFI's *imputation* mechanism, detailed in Appendix D.4.

Table 3: 5-D nonlinear regression with Gaussian noise. Mean (SE) over 100 replications.

Method	**MSE** ↓	**CI Cov. (95%)**	**CI Width** ↓	**PI Cov. (90%)**	**PI Width** ↓
EFI	0.0014 (0.0000)	0.9617 (0.0002)	0.1490 (0.0002)	0.8848 (0.0002)	0.3371 (0.0002)
SS-ANOVA	0.0012 (0.0001)	0.8972 (0.0054)	0.1011 (0.0009)	0.9050 (0.0018)	0.3677 (0.0025)
PINN (λ_{str}=10, dropout 1%)	0.0095 (0.0002)	0.9991 (0.0001)	0.8587 (0.0022)	0.9928 (0.0003)	0.7971 (0.0022)
PINN (λ_{str}=10) + SCP	0.0028 (0.0001)	–	–	0.9049 (0.0019)	0.3772 (0.0023)
Deep Ensemble (λ_{str}=100, M=50)	0.0008 (0.0000)	0.3521 (0.0048)	0.0258 (0.0002)	0.8788 (0.0014)	0.3224 (0.0010)



(a) True surface (b) EFI-learned surface (c) PINN-learned surface

Figure 3: x_1 and x_2 surface. See Figures D.8 and D.9 for more.

6 Conclusion and Discussion

Building on EFI, this paper develops a general inferential framework for dense neural networks under structural constraints by exploiting the low-dimensional structure of the admissible function class. When this class admits a locally identifiable low-dimensional representation, our theory justifies, under the stated regularity conditions, conducting inference in the reduced model and transferring the resulting uncertainty statements to identifiable functionals of the original dense model. These inferential guarantees do not require sparsity in the ambient network parameters. The narrow-neck EFI architecture provides a practical mechanism for realizing this low-dimensional representation while retaining a dense neural-network parameterization.

Our numerical experiments demonstrate the performance of the proposed framework across differential constraints, non-differential constraints, and higher-dimensional nonlinear regression: It provides calibrated uncertainty quantification for the latent regression target $f^\star$, while also producing prediction intervals for future responses Y^*, under both standard and nonstandard noise. These results have strong implications for scientific modeling with neural networks under suitable physical constraints, extending well beyond the traditional sparse-learning regime.

The main practical limitations are the availability of the structural information, the choice among competing structural model classes, and the computational cost of EFI sampling. When these restrictions are weak, misspecified, or overly restrictive, the resulting admissible class may no longer support well-calibrated or efficient inference. Future work includes diagnostics for constraint sufficiency, principled selection or refinement of structural restrictions, and more efficient sampling algorithms for large-scale constrained regression problems.

AI USE STATEMENT

In this work, we used generative AI tools to assist with refining mathematical statements and their assumptions, checking and revising proofs, and implementing the methods without changing the core algorithms. The proof assistance included suggestions for additional proof steps and corrections to arguments. We also used these tools for cluster job submission, manuscript editing, and converting CSV results into LaTeX tables. Assistance with the presentation of experimental results was limited to expressing interpretations already established by the authors. We did not use generative AI to generate synthetic datasets, develop theoretical models or conceptual frameworks, propose or refine research hypotheses, design or provide feedback on research methodology or experiments, translate text, clean or reformat datasets, conduct qualitative or thematic analysis, or interpret experimental results. The authors checked all AI-assisted mathematical statements and proofs, reviewed and tested the AI-assisted code, and take responsibility for the final content of the paper.

REFERENCES

Benedikt Bauer and Michael Kohler. On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *The Annals of Statistics*, 47(4):2261–2285, 2019.

Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. *The Annals of Statistics*, 38(5):2916 – 2957, 2010. doi: 10.1214/10-AOS799. URL https://doi.org/10.1214/10-AOS799.

Xiaowu Dai. Nonparametric estimation via partial derivatives. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 87(2):319–336, 2025.

Lutz Dümbgen. Shape-constrained statistical inference. *Annual review of statistics and its application*, 11, 2024.

D. A. S. Fraser. Structural probability and a generalization. *Biometrika*, 53:1–9, 1966.

D. A. S. Fraser. *The Structure of Inference*. John Wiley & Sons, New York-London-Sydney, 1968.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, volume PMLR 48, pp. 1050–1059, 2016. URL https://api.semanticscholar.org/CorpusID:160705.

Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992.

P. J. Green and B. W. Silverman. *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. Chapman and Hall, 1994. ISBN 0412300400.

Chong Gu. Smoothing spline anova models: R package gss. *Journal of Statistical Software*, 58:1–25, 2014.

Jan Hannig. On generalized fiducial inference. *Statistica Sinica*, 19(2):491–544, 2009.

Jan Hannig, Hari Iyer, Randy C. S. Lai, and Thomas C. M. Lee. Generalized fiducial inference: A review and new results. *Journal of the American Statistical Association*, 111(515):1346–1361, 2016. doi: 10.1080/01621459.2016.1165102.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2009.

Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical science*, 1(3):297–310, 1986.

Nancy E Heckman and James O Ramsay. Penalized regression with model-based penalties. *Canadian Journal of Statistics*, 28(2):241–258, 2000.

Sehwan Kim and Faming Liang. Extended fiducial inference for individual treatment effects via deep neural networks. *Statistics and Computing*, 35, 2025. URL https://api.semanticscholar.org/CorpusID:278326914.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Neural Information Processing Systems*, 2016. URL https://api.semanticscholar.org/CorpusID:6294674.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry A. Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113:1094 – 1111, 2018.

Faming Liang. *Deep Learning and Statistics: Toward Automated Inference and Uncertainty-Aware AI*. CRC Press, Boca Raton, FL, 2026.

Faming Liang, Qizhai Li, and Lei Zhou. Bayesian neural networks for selection of drug sensitive genes. *Journal of the American Statistical Association*, 113(523):955–972, 2018.

Faming Liang, Sehwan Kim, and Yan Sun. Extended fiducial inference: Toward an automated process of statistical inference. *Journal of the Royal Statistical Society Series B*, 87:98–131, 2025.

Siqi Liang, Yan Sun, and Faming Liang. Nonlinear sufficient dimension reduction with a stochastic neural network. *NeurIPS 2022*, 2022.

Y. Lin and H.H. Zhang. Component selection and smoothing in multivariate nonparametric regression. *Annals of Statistics*, 34:2272–2297, 2006.

Catherine Loader and Maintainer Andy Liaw. Package 'locfit', 2025.

Ryan Martin. Fiducial inference viewed through a possibility-theoretic inferential model lens. *Journal of Machine Learning Research*, 215:299–310, 2023. URL https://api.semanticscholar.org/CorpusID:257532597.

Ryan Martin and Chuanhai Liu. Inferential models: A framework for prior-free posterior probabilistic inference. *Journal of the American Statistical Association*, 108:301 – 313, 2013.

Ryan Martin and Chuanhai Liu. *Inferential Models: Reasoning with Uncertainty*. CRC Press, 2015.

J. Milnor and James D. Stasheff. *Characteristic Classes*. Princeton University Press, 1974. URL https://api.semanticscholar.org/CorpusID:124155846.

Alexander C. Murph, Jan Hannig, and Jonathan P. Williams. Generalized fiducial inference on differentiable manifolds. *arXiv:2209.15473*, 2022. URL https://api.semanticscholar.org/CorpusID:252668395.

Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019. URL https://api.semanticscholar.org/CorpusID:57379996.

P. Ravikumar, J. Lafferty, H. Liu, and L. Wasserman. Sparse additive models. *Journal of the Royal Statistical Society, Series B*, 71:1009–1030, 2009.

Thomas J Rothenberg. Identification in parametric models. *Econometrica: Journal of the Econometric Society*, pp. 577–591, 1971.

Johannes Schmidt-Hieber. Nonparametric regression using deep neural networks with relu activation function. *Annals of Statistics*, 48(4):1875–1897, 2020.

Frank Shih, Zhenghao Jiang, and Faming Liang. Uncertainty quantification for physics-informed neural networks with extended fiducial inference. *NeurIPS*, 2025.

Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

Yan Sun, Qifan Song, and Faming Liang. Consistent sparse deep learning: Theory and computation. *Journal of the American Statistical Association*, 117(540):1981–1995, 2022.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

Grace Wahba. *Splines models for observational data*. SIAM, Philadelphia, 1990.

Grace Wahba, Yuedong Wang, Chong Gu, Ronald Klein, and Barbara Klein. Smoothing spline anova for exponential families, with application to the wisconsin epidemiological study of diabetic retinopathy: the 1994 neyman memorial lecture. *The Annals of Statistics*, 23(6):1865–1895, 1995.

Wen-Hung Wang and Faming Liang. Extended fiducial inference for statistical models with unknown noise distributions. *Manuscript (submitted for publication)*, 2026.

L. Wasserman. *All of Nonparametric Statistics*. Springer Texts in Statistics. Springer, New York, 2006. ISBN 978-0-387-25145-5. doi: 10.1007/0-387-30623-4.

Simon Wood and Maintainer Simon Wood. Package 'mgcv'. *R package version*, 1(29):729, 2015.

Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2020.109913. URL https://www.sciencedirect.com/science/article/pii/S0021999120306872.

Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *J. Comput. Phys.*, 397:108850, 2019. URL https://api.semanticscholar.org/CorpusID:53368962.

APPENDIX OVERVIEW

This appendix supplements the main text with full theoretical details, methodological background, and extended experimental results. Appendix A introduces the notations used in the main text. Appendix B collects all proofs for the theoretical results stated in Section 4, including the narrow-neck verification lemma and the proofs of Theorem 4.3 and Corollary 4.4. Appendix C provides a self-contained review of the EFI and NE-EFI methods, covering the energy function formulation, the adaptive SGLD algorithm, the extended fiducial density, and convergence diagnostics. Appendix D presents the full simulation results corresponding to Section 5, with detailed tables and figures for the 1-D Poisson equation (bimodal and Gaussian noise settings) and the periodic and 5-D nonlinear regression examples. Appendix E examines sensitivity to the consistency coefficient, KDE bandwidth, SGLD schedule, and temperature. Appendix F reports additional experiments—a 1-D linear model with skewed errors, the Liang et al. (2018) high-dimensional example, and a monotonic regression—that further stress-test the framework's flexibility and robustness. Finally, Appendix G documents all hyperparameter configurations used across experiments to facilitate reproducibility.

A NOTATION

The table below summarizes the mathematical symbols used in the main text (§1–§6). Appendix-only notation introduced in the proofs and supplementary derivations is defined locally where it appears.

Symbol	Meaning / first appearance
Data, indices, sample	
n	sample size
$\boldsymbol{x}_i \in \mathbb{R}^d$	ith covariate (d = covariate dimension)
$y_i \in \mathbb{R}$	ith response
$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$	observed data; §1 eq. (1)
Y^*	future noisy response (PI target); §5, §6
i, j, ℓ	sample / constraint-evaluation / constraint-family indices
Function classes and estimators	
$f^\star$	true target regression function; §1 eq. (1)
$\mathcal{F}$	generic restricted function class; §1 eq. (2)
$f_{\boldsymbol{\theta}}(\boldsymbol{x})$	dense-network parameterized regression function; §1 eq. (3) onward
Parameters (body)	
$\boldsymbol{\theta} \in \mathbb{R}^p$	dense network parameter; §1, §2, §3
$\hat{\boldsymbol{\theta}}_i = G_{\boldsymbol{w}}(\boldsymbol{x}_i, y_i, z_i)$	per-sample dense parameter from the EFI inverse map; §3
$\bar{\boldsymbol{\theta}}_n = (1/n)\sum_i \hat{\boldsymbol{\theta}}_i$	sample average dense parameter; §3
Latent noise and distributions	
z_i	per-sample latent noise variable
$\boldsymbol{Z}_n = (z_1, \ldots, z_n)$	full latent noise vector
$\hat{\boldsymbol{Z}}_n^{(t)}$	imputed latent samples at refinement run t; §3, Fig. 2, Algorithm
π_0	reference noise law (may be known or unknown)
$\pi^{(t)}$	working noise distribution at refinement run t; Fig. 2, Algorithm
σ (σ^2)	observational noise scale (resp. variance); jointly estimated with $\boldsymbol{\theta}$ in EFI; Algorithm, §5
Constraints, penalties, residual functionals	
$\mathfrak{C}(f)$	bundled structural restriction; §1 eq. (1), §4 Assumption 4.1
$\mathcal{R}_{\mathrm{sm}}(f)$	smoothness regularizer; §1 eq. (2), §2
$\mathcal{R}_{\mathrm{str}}(f_{\boldsymbol{\theta}})$	generic structural penalty term; §1 eqs. (3), (4), §2
λ_{str}	structural-penalty weight in PINN-style criteria; §1, §2
$\eta_{\mathrm{str}}, \eta_\theta$	EFI structural-term and consistency-term computational scalings; §3 eq. (7), §5 (Table 1 rows)
$C^{(\ell)}(f; \tilde{\boldsymbol{x}})$	ℓth structural constraint operator; §3 eq. (6)
L, m_ℓ	number of constraint families / evaluation points per family; §3
$\tilde{\boldsymbol{x}}_j^{(\ell)}$	jth evaluation point of constraint family ℓ; §3
$V^{(\ell)} : \mathbb{R} \to \mathbb{R}_{\geq 0}$	violation function: c^2 for $\ell \in \mathcal{E}$, $\mathrm{ReLU}(c)^2$ for $\ell \in \mathcal{I}$; §3
$\mathcal{E}, \mathcal{I}$	equality / inequality partition of $\{1, \ldots, L\}$; §3
σ_{diff}	BPINN-specific differential-residual scale (*not* the observational σ); §5, Table 2
EFI machinery (§3)	
w	inverse-map neural-network weights
$G_{\boldsymbol{w}}$	learned inverse map $(\boldsymbol{x}, y, z) \mapsto \boldsymbol{\theta}$ at weights w
$U(\boldsymbol{\theta}; \mathcal{D}, \boldsymbol{Z}_n)$ / $U(w; \mathcal{D}, \boldsymbol{Z}_n)$	EFI energy in dense-parameter / inverse-map form; §1 eq. (4), §3 eq. (7)
ϵ	MCMC temperature in $\exp\{-U/\epsilon\}$
Theoretical reduction (§4)	
$\mathcal{X} \subseteq \mathbb{R}^d$	covariate space
$\Theta \subseteq \mathbb{R}^p$	dense network parameter space
$\Theta_c := \{\boldsymbol{\theta} \in \Theta : \mathfrak{C}(f_{\boldsymbol{\theta}}) = 0\}$	constrained parameter set; Assumption 4.1
$\mathcal{F}_c = \{f_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta_c\}$	constrained function class
$\Phi \subseteq \mathbb{R}^q$	low-dim effective parameter space ($q \ll p$, $q < n$)
$\Phi_c := \rho(\Theta_c) \subseteq \Phi$	feasible reduced parameter set
$\rho : \Theta_c \to \Phi$	constrained-manifold reduction map; $\boldsymbol{\varphi} := \rho(\boldsymbol{\theta})$
$g : \mathcal{X} \times \Phi \to \mathbb{R}$	reduced regression function: $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = g(\boldsymbol{x}, \rho(\boldsymbol{\theta}))$
$\boldsymbol{\varphi} \in \Phi$	low-dim effective parameter
$\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*$	true (data-generating) dense / reduced parameter; $\boldsymbol{\varphi}^* := \rho(\boldsymbol{\theta}^*)$
$b : \Phi_c \to \mathbb{R}$	generic identifiable functional of the reduced parameter; Corollary 4.4
Simulation baselines (§5)	
M	deep-ensemble size; Tables 1, 2, 3

B PROOFS FOR SECTION 4

The following paragraph first reviews the narrow-neck EFI construction of Shih et al. (2025) and uses it to obtain a low-dimensional representation of the EFI-restricted class. This representation is distinct from the reduction of the entire constrained class postulated in Assumption 4.1. Lemma B.1 then provides a sufficient Jacobian rank condition for local identifiability of the resulting low-dimensional family. Finally, we prove Theorem 4.3 and Corollary 4.4.

Narrow-neck representation. Let

$$\mathcal{Z}_n = \{\boldsymbol{z} \in \mathbb{R}^n : U(w_n^*; \mathcal{D}, \boldsymbol{z}) = 0\}, \qquad \hat{\boldsymbol{\theta}}(\boldsymbol{z}) = G_{w_n^*}(\boldsymbol{z}, \mathcal{D}),$$

where $\boldsymbol{w}_n^*$ denotes a solution to equation (C.7) and

$$\mathcal{F}_n^{\mathrm{EFI}} = \{f_{\hat{\boldsymbol{\theta}}(\boldsymbol{z})} : \boldsymbol{z} \in \mathcal{Z}_n\}$$

where $\hat{\boldsymbol{\theta}}(\boldsymbol{z})$ denotes a generic EFI estimator for $\boldsymbol{\theta}$, e.g., $\bar{\boldsymbol{\theta}}_n$ in the main text.

Assume that the learned inverse map admits the narrow-neck representation

$$\hat{\boldsymbol{\theta}}(\boldsymbol{z}) = \mathbf{a}_n + \Xi_n^\top \boldsymbol{m}(\boldsymbol{z}), \qquad \boldsymbol{m}(\boldsymbol{z}) \in \mathbb{R}^{d_h}, \tag{B.1}$$

where $\mathbf{a}_n \in \mathbb{R}^p$ and $\Xi_n \in \mathbb{R}^{d_h \times p}$ are fixed, $d_h < n$, $\boldsymbol{m}(\boldsymbol{z})$ is the output of the last hidden layer of the $w-$network, i.e. the neck layer (Figure C.2) and $d_h/n \to 0$. Note that the dependence of d_h on n is implicit and that the linear relationship $\hat{\boldsymbol{\theta}} \sim \boldsymbol{m}$ can be generally ensured by the universal approximation ability of the DNN as noted by Shih et al. (2025). Additionally, we note that the affine representation (B.1) can be implemented using the narrow-neck $\boldsymbol{w}$-network developed by Shih et al. (2025) and it can be realized at the zero-energy solution.

Next, we define

$$\mathcal{T}_n := \{\boldsymbol{m}(\boldsymbol{z}) : \boldsymbol{z} \in \mathcal{Z}_n\} \subseteq \mathbb{R}^{d_h}$$

and

$$g_n(x, \boldsymbol{\tau}) := f_{\mathbf{a}_n + \Xi_n^\top \boldsymbol{\tau}}(x), \qquad \boldsymbol{\tau} \in \mathcal{T}_n.$$

Hence $\mathcal{F}_n^{\mathrm{EFI}}$ admits an exact representation by the low-dimensional family

$$\mathcal{G}_n := \{g_n(\cdot, \boldsymbol{\tau}) : \boldsymbol{\tau} \in \mathcal{T}_n\}, \qquad \mathcal{T}_n \subseteq \mathbb{R}^{d_h}.$$

As discussed in Section 4, the remaining gap is whether the EFI function class $\mathcal{F}_n^{\mathrm{EFI}}$ is sufficiently rich to approximate the admissible class $\mathcal{F}_c$, in the sense that any function in $\mathcal{F}_c$ can be well approximated by a function in $\mathcal{F}_n^{\mathrm{EFI}}$. In practice, the adequacy of this approximation can be assessed by increasing the bottleneck dimension d_h and examining the resulting sensitivity.

Lemma B.1 (Verifying local identifiability)**.** *Suppose that (B.1) holds and that, for every $\boldsymbol{\tau}_0 \in \mathcal{T}_n$, the same formula for g_n is defined on an open neighborhood $O_{\boldsymbol{\tau}_0} \subseteq \mathbb{R}^{d_h}$ of $\boldsymbol{\tau}_0$, and there exist points $u_1, \ldots, u_{d_h} \in \mathcal{X}$ such that the evaluation map*

$$F_n(\boldsymbol{\tau}) := \big(g_n(u_1, \boldsymbol{\tau}), \ldots, g_n(u_{d_h}, \boldsymbol{\tau})\big)^\top$$

is continuously differentiable on $O_{\boldsymbol{\tau}_0}$ and satisfies

$$J_n(\boldsymbol{\tau}_0) := \left.\frac{\partial F_n(\boldsymbol{\tau})}{\partial \boldsymbol{\tau}^\top}\right|_{\boldsymbol{\tau}=\boldsymbol{\tau}_0}, \qquad \operatorname{rank}\big(J_n(\boldsymbol{\tau}_0)\big) = d_h.$$

Then, for every $\boldsymbol{\tau}_0 \in \mathcal{T}_n$, there exists an open neighborhood $V_{\boldsymbol{\tau}_0} \subseteq O_{\boldsymbol{\tau}_0}$ of $\boldsymbol{\tau}_0$ such that

$$g_n(\cdot, \boldsymbol{\tau}_1) = g_n(\cdot, \boldsymbol{\tau}_2), \qquad \boldsymbol{\tau}_1, \boldsymbol{\tau}_2 \in \mathcal{T}_n \cap V_{\boldsymbol{\tau}_0} \quad \Longrightarrow \quad \boldsymbol{\tau}_1 = \boldsymbol{\tau}_2.$$

Consequently, the the low-dimensional family $\mathcal{G}_n$ is locally identifiable on $\mathcal{T}_n$.

Proof. To establish local identifiability, fix $\boldsymbol{\tau}_0 \in \mathcal{T}_n$ and choose the evaluation points and neighborhood. Since F_n is continuously differentiable and its square Jacobian $J_n(\boldsymbol{\tau}_0)$ is nonsingular, the inverse function theorem gives an open neighborhood $V_{\boldsymbol{\tau}_0} \subseteq O_{\boldsymbol{\tau}_0}$ on which F_n is one-to-one. If $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2 \in \mathcal{T}_n \cap V_{\boldsymbol{\tau}_0}$ and $g_n(\cdot, \boldsymbol{\tau}_1) = g_n(\cdot, \boldsymbol{\tau}_2)$, equality at the selected evaluation points implies

$F_n(\boldsymbol{\tau}_1) = F_n(\boldsymbol{\tau}_2)$, and hence $\boldsymbol{\tau}_1 = \boldsymbol{\tau}_2$. Since $\boldsymbol{\tau}_0$ was arbitrary, the low-dimensional family $\mathcal{G}_n$ is locally identifiable on $\mathcal{T}_n$.

At any point where the Jacobian rank condition holds, it also implies $\operatorname{rank}(\Xi_n) = d_h$. To see this, if $\Xi_n^\top v = 0$, then $F_n(\boldsymbol{\tau}_0 + tv) = F_n(\boldsymbol{\tau}_0)$ for all sufficiently small t. Differentiating at $t = 0$ gives $J_n(\boldsymbol{\tau}_0)v = 0$, so $v = 0$. Thus a separate full-row-rank assumption on Ξ_n is unnecessary. □

Remark B.2. Such Jacobian rank conditions are classical sufficient conditions for local identifiability in nonlinear parametric models. For example, Rothenberg (1971) established, under regularity conditions, that local identifiability is equivalent to nonsingularity of the information matrix.

Proof of Theorem 4.3. Fix $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and $\boldsymbol{Z}_n = (z_1, \ldots, z_n)$. Define

$$S_n^\theta(\mathcal{D}, \boldsymbol{Z}_n) := \{\boldsymbol{\theta} \in \Theta_c : y_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + z_i, \;\; i = 1, \ldots, n\},$$

and

$$S_n^\varphi(\mathcal{D}, \boldsymbol{Z}_n) := \{\boldsymbol{\varphi} \in \Phi_c : y_i = g(\boldsymbol{x}_i, \boldsymbol{\varphi}) + z_i, \;\; i = 1, \ldots, n\}.$$

We first show that

$$S_n^\varphi(\mathcal{D}, \boldsymbol{Z}_n) = \rho\big(S_n^\theta(\mathcal{D}, \boldsymbol{Z}_n)\big). \tag{B.2}$$

If $\boldsymbol{\theta} \in S_n^\theta(\mathcal{D}, \boldsymbol{Z}_n)$, then by (8),

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = g(\boldsymbol{x}_i, \rho(\boldsymbol{\theta})), \qquad i = 1, \ldots, n,$$

and therefore

$$y_i = g(\boldsymbol{x}_i, \rho(\boldsymbol{\theta})) + z_i, \qquad i = 1, \ldots, n.$$

Hence $\rho(\boldsymbol{\theta}) \in S_n^\varphi(\mathcal{D}, \boldsymbol{Z}_n)$, so

$$\rho\big(S_n^\theta(\mathcal{D}, \boldsymbol{Z}_n)\big) \subseteq S_n^\varphi(\mathcal{D}, \boldsymbol{Z}_n).$$

Conversely, let $\boldsymbol{\varphi} \in S_n^\varphi(\mathcal{D}, \boldsymbol{Z}_n)$. Since $\Phi_c = \rho(\Theta_c)$, there exists $\boldsymbol{\theta} \in \Theta_c$ such that $\rho(\boldsymbol{\theta}) = \boldsymbol{\varphi}$. Again by (8),

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = g(\boldsymbol{x}_i, \boldsymbol{\varphi}), \qquad i = 1, \ldots, n.$$

Because $\boldsymbol{\varphi} \in S_n^\varphi(\mathcal{D}, \boldsymbol{Z}_n)$,

$$y_i = g(\boldsymbol{x}_i, \boldsymbol{\varphi}) + z_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + z_i, \qquad i = 1, \ldots, n,$$

so $\boldsymbol{\theta} \in S_n^\theta(\mathcal{D}, \boldsymbol{Z}_n)$. Therefore

$$S_n^\varphi(\mathcal{D}, \boldsymbol{Z}_n) \subseteq \rho\big(S_n^\theta(\mathcal{D}, \boldsymbol{Z}_n)\big),$$

which proves (B.2).

Now define the latent compatibility sets

$$\mathcal{Z}_n^\theta(\mathcal{D}) := \{\boldsymbol{z} \in \mathbb{R}^n : S_n^\theta(\mathcal{D}, \boldsymbol{z}) \neq \emptyset\}, \qquad \mathcal{Z}_n^\varphi(\mathcal{D}) := \{\boldsymbol{z} \in \mathbb{R}^n : S_n^\varphi(\mathcal{D}, \boldsymbol{z}) \neq \emptyset\}.$$

By (B.2),

$$S_n^\theta(\mathcal{D}, \boldsymbol{z}) \neq \emptyset \iff S_n^\varphi(\mathcal{D}, \boldsymbol{z}) \neq \emptyset,$$

and hence

$$\mathcal{Z}_n^\theta(\mathcal{D}) = \mathcal{Z}_n^\varphi(\mathcal{D}).$$

Thus the dense and reduced formulations determine the same EFI zero-energy set in the latent variable space. □

Proof of Corollary 4.4. By Theorem 4.3, the dense and reduced formulations have the same feasible latent set. Moreover, as shown in the proof of Theorem 4.3, for any $b : \Phi_c \to \mathbb{R}$, the two formulations induce the same feasible values, namely,

$$\{b(\boldsymbol{\varphi}) : \boldsymbol{\varphi} \in S_n^\varphi(\mathcal{D}, \boldsymbol{Z}_n)\} = \{b(\rho(\boldsymbol{\theta})) : \boldsymbol{\theta} \in S_n^\theta(\mathcal{D}, \boldsymbol{Z}_n)\}.$$

Hence, if EFI applies to the reduced model for inference on the quantity of interest $b(\boldsymbol{\varphi}^*)$, then the same latent compatibility relation induces the corresponding EFI uncertainty statement for the dense-model quantity $b(\rho(\boldsymbol{\theta}^*))$. □

C REVIEW OF THE EFI AND NE-EFI METHODS

C.1 THE EFI METHOD

While fiducial inference was widely considered as a big blunder by R.A. Fisher, the goal he initially set —inferring the uncertainty of model parameters based solely on observations — has been continually pursued by many statisticians, see e.g. structural inference (Fraser, 1966; 1968), generalized fiducial inference (Hannig, 2009; Hannig et al., 2016; Murph et al., 2022), and inferential models (Martin & Liu, 2013; 2015; Martin, 2023). To this end, Liang et al. (2025) developed the EFI method based on the fundamental concept of structural inference.

Consider a regression model:

$$Y = f(\boldsymbol{X}, Z, \boldsymbol{\theta}), \tag{C.1}$$

where $Y \in \mathbb{R}$ and $\boldsymbol{X} \in \mathbb{R}^d$ represent the response and explanatory variables, respectively; $\boldsymbol{\theta} \in \mathbb{R}^p$, with $p \leq n$, 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 $\{(y_1, \boldsymbol{x}_1), (y_2, \boldsymbol{x}_2), \ldots, (y_n, \boldsymbol{x}_n)\}$, has been collected from the model. In structural inference, the observations can be expressed in data-generating equations as follows:

$$y_i = f(\boldsymbol{x}_i, z_i, \boldsymbol{\theta}), \quad i = 1, 2, \ldots, n. \tag{C.2}$$

This system of equations consists of $n + p$ unknowns, namely, $\{\boldsymbol{\theta}, z_1, z_2, \ldots, z_n\}$, 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 $\boldsymbol{Z}_n = \{z_1, z_2, \ldots, z_n\}$ 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.,

$$\boldsymbol{\theta} = G(\boldsymbol{Y}_n, \boldsymbol{X}_n, \boldsymbol{Z}_n). \tag{C.3}$$

As noted by Liang et al. (2025), this non-uniqueness of inverse function mirrors the flexibility of frequentist methods, where different estimators of $\boldsymbol{\theta}$ can be constructed to achieve desired properties such as efficiency, unbiasedness, and robustness.

Since the inverse function $G(\cdot)$ is generally unknown, Liang et al. (2025) proposed to approximate it using a sparse DNN, while Shih et al. (2025) proposed to approximate it using a narrow-neck DNN, where the dimension of the last hidden layer $d_h \prec n$, both for ensuring consistency. For both of them, the so-called EFI-network can be illustrated by Figures C.1 and C.2.

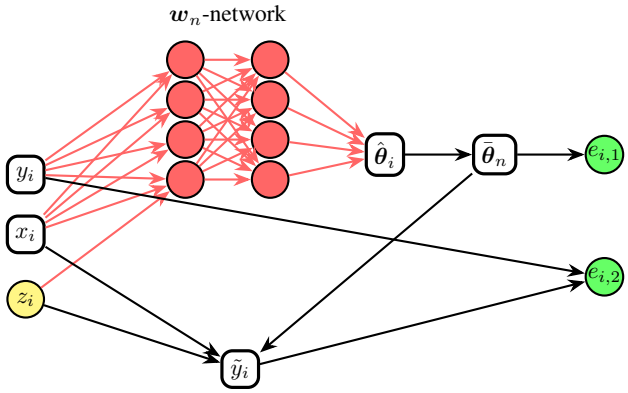


Figure C.1: Illustration of the EFI-network in low-dimensional settings, i.e, f is not a DNN. The red nodes and links form a DNN (with parameters $\boldsymbol{w}_n$) that learns the inverse mapping; the yellow node represents latent variables to be imputed; the green nodes represent two types of output errors; and black arrows denote deterministic relationships.

Briefly, they let $\boldsymbol{w}_n$ denote the weights of $\boldsymbol{w}_n$-network and let $g(y, \boldsymbol{x}, z, \boldsymbol{w}_n)$ be the corresponding neural network output function. They define an energy function:

$$U_n(\boldsymbol{Y}_n, \boldsymbol{X}_n, \boldsymbol{Z}_n, \boldsymbol{w}_n) = \eta \sum_{i=1}^{n} \|e_{i,1}\|^2 + \sum_{i=1}^{n} \|e_{i,2}\|^2 \tag{C.4}$$

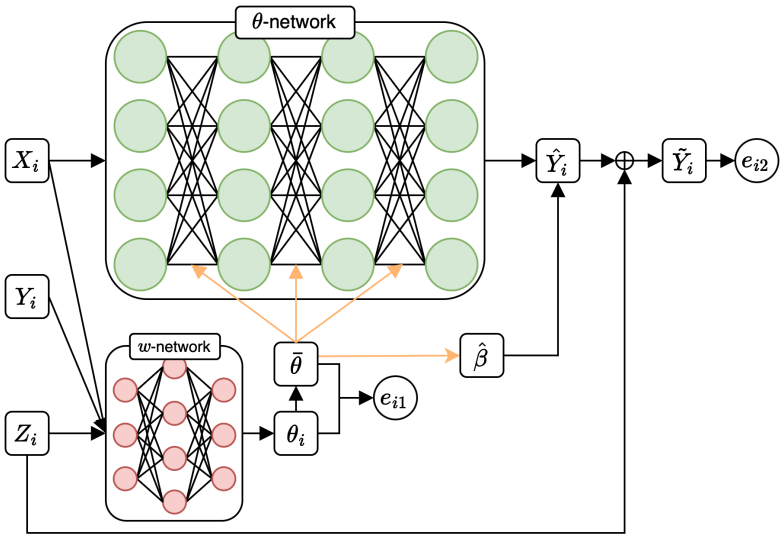


Figure C.2: EFI-network architecture used in Shih et al. (2025) for estimating DNNs, i.e., f is a DNN parameterized by θ (θ-network); while the $w-$network learns the inverse mapping. A *narrow-neck* $w-$network means the dimension of the last hidden layer $d_h \prec n$. Here $\hat{\beta}$ denotes the parameter of interest; in the present paper, this may include the observation noise scale σ in addition to neural-network parameters θ.

where $\hat{\boldsymbol{\theta}}_i = g(y_i, \boldsymbol{x}_i, z_i, \boldsymbol{w}_n)$, $\bar{\boldsymbol{\theta}}_n = \frac{1}{n}\sum_{i=1}^n \hat{\boldsymbol{\theta}}_i$, $e_{i,1} = \hat{\boldsymbol{\theta}}_i - \bar{\boldsymbol{\theta}}_n$, $e_{i,2} = y_i - f(\boldsymbol{x}_i, z_i, \bar{\boldsymbol{\theta}}_n)$, and η is a regularization parameter. Accordingly, they define

$$\pi_\epsilon(\boldsymbol{Y}_n|\boldsymbol{X}_n, , \boldsymbol{Z}_n, \boldsymbol{w}_n) \propto e^{-U_n(\boldsymbol{Y}_n, \boldsymbol{X}_n, \boldsymbol{Z}_n, \boldsymbol{w}_n)/\epsilon}, \tag{C.5}$$

for some constant ϵ close to 0, where ϵ can be read as a temperature parameter. As discussed in Liang et al. (2025) , the choice of η does not affect much on the performance of EFI as long as ϵ is sufficiently small. Subsequently, a predictive distribution for $\boldsymbol{Z}_n$ can be defined as:

$$\pi_\epsilon(\boldsymbol{Z}_n|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n) \propto \pi_0^{\otimes n}(\boldsymbol{Z}_n) e^{-U_n(\boldsymbol{Y}_n, \boldsymbol{X}_n, \boldsymbol{Z}_n, \boldsymbol{w}_n)/\epsilon} \tag{C.6}$$

where $\pi_0^{\otimes n}(\boldsymbol{Z}_n) = \prod_{i=1}^n \pi_0(z_i)$ under the assumption that the z_i are independent and identically distributed (i.i.d.). In EFI, $\boldsymbol{Z}_n$ is treated as missing data. Shih et al. (2025) applied an adaptive SGLD algorithm (Algorithm C.1) to learn $\boldsymbol{w}_n$ through solving the following equation:

$$\nabla_{\boldsymbol{w}_n} \log \pi_\epsilon(\boldsymbol{Y}_n|\boldsymbol{X}_n, \boldsymbol{w}_n) = \int \Big[\nabla_{\boldsymbol{w}_n} \log \pi_\epsilon(\boldsymbol{Y}_n|\boldsymbol{X}_n, \boldsymbol{Z}_n, \boldsymbol{w}_n) \pi_\epsilon(\boldsymbol{Z}_n|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n)\Big] d\boldsymbol{Z}_n = 0, \tag{C.7}$$

where the first equality follows from Fisher's identity. Refer to Liang et al. (2025) for a Bayesian version of Algorithm C.1, where a mixture-Gaussian prior is imposed on $\boldsymbol{w}_n$ to ensure its sparsity.

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

$$\|\boldsymbol{w}_n^{(k)} - \boldsymbol{w}_n^*\| \xrightarrow{p} 0, \quad \text{as } k \to \infty, \tag{C.10}$$

where $\boldsymbol{w}_n^*$ denotes a solution to equation (C.7) and $\xrightarrow{p}$ denotes convergence in probability, and that

$$\boldsymbol{Z}_n^{(k)} \overset{d}{\rightsquigarrow} \pi_\epsilon(\boldsymbol{Z}_n|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n^*), \quad \text{as } k \to \infty, \tag{C.11}$$

in 2-Wasserstein distance, where $\overset{d}{\rightsquigarrow}$ denotes weak convergence. To study the limit of (C.11) as ϵ decays to 0, i.e., $p_n^*(\boldsymbol{z}|\boldsymbol{Y}_n, \boldsymbol{X}_n, \boldsymbol{w}_n^*) = \lim_{\epsilon \downarrow 0} \pi_\epsilon(\boldsymbol{Z}_n|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n^*)$, where $p_n^*(\boldsymbol{z}|\boldsymbol{Y}_n, \boldsymbol{X}_n, \boldsymbol{w}_n^*)$ is referred to as the extended fiducial density (EFD) of $\boldsymbol{Z}_n$, Liang et al. (2025) impose specific conditions on the structure of the $\boldsymbol{w}_n$-network, including that the $\boldsymbol{w}_n$-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

Algorithm C.1: Adaptive SGLD for EFI computation

(i) (Initialization) Initialize $\boldsymbol{w}_n^{(0)}$, $\boldsymbol{Z}_n^{(0)}$, M (the number of fiducial samples to collect), and $\mathcal{K}$ (burn-in iterations).
for *k=0,1,...,*$\mathcal{K}+M-1$ **do**
  (ii) (Latent variable imputation) Given $\boldsymbol{w}_n^{(k)}$, simulate $\boldsymbol{Z}_n^{(k+1)}$ using the SGLD algorithm:

$$\boldsymbol{Z}_n^{(k+1)} = \boldsymbol{Z}_n^{(k)} + \upsilon_{k+1}\nabla_{\boldsymbol{Z}_n}\log\pi_\epsilon(\boldsymbol{Z}_n^{(k)}|\boldsymbol{X}_n,\boldsymbol{Y}_n,\boldsymbol{w}_n^{(k)}) + \sqrt{2\upsilon_{k+1}}\boldsymbol{e}^{(k+1)} \tag{C.8}$$

  where υ_{k+1} is the learning rate, and $\boldsymbol{e}^{(k+1)} \sim \mathcal{N}(0, I_{d_{\boldsymbol{z}}})$.
  (iii) (Parameter updating) Update the network weights by the SGD algorithm:

$$\boldsymbol{w}_n^{(k+1)} = \boldsymbol{w}_n^{(k)} + \gamma_{k+1}\nabla_{\boldsymbol{w}_n}\log\pi(\boldsymbol{Y}_n|\boldsymbol{X}_n,\boldsymbol{Z}_n^{(k+1)},\boldsymbol{w}_n^{(k)}), \tag{C.9}$$

  where γ_{k+1} is the step size.
  (iv) (Fiducial sample collection) If $k+1 > \mathcal{K}$, calculate $\hat{\boldsymbol{\theta}}_i^{(k+1)} = g(y_i, \boldsymbol{x}_i, z_i^{(k+1)}, \boldsymbol{w}_n^{(k+1)})$ for each $i \in \{1,2,\ldots,n\}$ and average them to get a fiducial $\bar{\boldsymbol{\theta}}_n$-sample calculated as in (C.4).
end
(v) (Statistical Inference) Conducting statistical inference for the model based on the collected fiducial samples.

n. Under these assumptions, they prove the consistency of $\boldsymbol{w}_n^*$ based on the sparse deep learning theory developed in Sun et al. (2022). This consistency further implies that

$$\widehat{G}(\boldsymbol{Y}_n,\boldsymbol{X}_n,\boldsymbol{Z}_n) = \frac{1}{n}\sum_{i=1}^{n} g(y_i,\boldsymbol{x}_i,z_i,\boldsymbol{w}_n^*), \tag{C.12}$$

serves as a consistent estimator for the inverse function/mapping $\boldsymbol{\theta} = G(\boldsymbol{Y}_n,\boldsymbol{X}_n,\boldsymbol{Z}_n)$, where $g(\cdot)$ denotes the learned neural network function. Refer to Appendix C.2 for the expression of $p_n^*(\boldsymbol{z}|\boldsymbol{Y}_n,\boldsymbol{X}_n,\boldsymbol{w}_n^*)$.

Constraining the dimension of $\boldsymbol{\theta}$ to be fixed or to grow slowly with the sample size n ensures the consistency of the estimator in (C.12), but such an assumption is typically violated in inference for high-dimensional neural network models. To address this issue, Shih et al. (2025) propose an architecture for the $\boldsymbol{w}_n$-network, in which the width d_h of the last hidden layer is sufficiently small. They establish consistency of the resulting inverse mapping by leveraging the sufficient dimension reduction (SDR) property of the Stochastic Network (StoNet) (Liang et al., 2022) together with prediction consistency for low-dimensional linear regression. That said, the narrow-network regime sidesteps a Bayesian treatment of learning the inverse mapping. In terms of applications, Shih et al. (2025) extend this approach to uncertainty quantification for physics-informed neural networks (PINNs), where the network outputs are constrained by physical laws.

Let $\mathcal{Z}_n = \{\boldsymbol{z} \in \mathbb{R}^n : U_n(\boldsymbol{Y}_n,\boldsymbol{X}_n,\boldsymbol{Z}_n,\boldsymbol{w}_n^*) = 0\}$ 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 $\Theta := \{\boldsymbol{\theta} \in \mathbb{R}^p : \boldsymbol{\theta} = G(\boldsymbol{Y}_n,\boldsymbol{X}_n,\boldsymbol{z}), \boldsymbol{z} \in \mathcal{Z}_n\}$ denote the parameter space of the target model, which represents the set of all possible values of $\boldsymbol{\theta}$ that $G(\cdot)$ takes when $\boldsymbol{z}$ runs over $\mathcal{Z}_n$. Then, for any function $b(\boldsymbol{\theta})$ of interest, its EFD $\mu_n^*(\cdot|\boldsymbol{Y}_n,\boldsymbol{X}_n)$ associated with $G(\cdot)$ is given by

$$\mu_n^*(B|\boldsymbol{Y}_n,\boldsymbol{X}_n) = \int_{\mathcal{Z}_n(B)} dP_n^*(\boldsymbol{z}|\boldsymbol{Y}_n,\boldsymbol{X}_n,\boldsymbol{w}_n^*), \tag{C.13}$$

for any measurable set $B \subset b(\Theta)$, where $\mathcal{Z}_n(B) = \{\boldsymbol{z} \in \mathcal{Z}_n : b(G(\boldsymbol{Y}_n,\boldsymbol{X}_n,\boldsymbol{z})) \in B\}$, and $P_n^*(\boldsymbol{z}|\boldsymbol{X}_n,\boldsymbol{Y}_n,\boldsymbol{w}_n^*)$ denote the cumulative distribution function (CDF) corresponding to $p_n^*(\boldsymbol{z}|\boldsymbol{X}_n,\boldsymbol{Y}_n,\boldsymbol{w}_n^*)$. The EFD provides an uncertainty measure for $b(\boldsymbol{\theta})$. Practically, it can be constructed based on the samples $\{b(\bar{\boldsymbol{\theta}}_n^{(1)}), b(\bar{\boldsymbol{\theta}}_n^{(2)}), \ldots, b(\bar{\boldsymbol{\theta}}_n^{(M)})\}$, where $\{\bar{\boldsymbol{\theta}}_n^{(1)}, \bar{\boldsymbol{\theta}}_n^{(2)}, \ldots, \bar{\boldsymbol{\theta}}_n^{(M)}\}$ denotes the fiducial $\bar{\boldsymbol{\theta}}$-samples collected at step (iv) of Algorithm C.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 $\boldsymbol{w}$-network, the consistency of $\boldsymbol{w}_n^*$ refers to its consistency with respect to one of the equivalent solutions to (C.7), while mathematically $\boldsymbol{w}_n^*$ can still be treated as unique.

C.2 Extended Fiducial Density Function $\boldsymbol{Z}_n$

Let $\mathcal{Z}_n = \{\boldsymbol{z} \in \mathbb{R}^n : U_n(\boldsymbol{Y}_n, \boldsymbol{X}_n, \boldsymbol{Z}_n, \boldsymbol{w}_n^*) = 0\}$ denote the zero-energy set, and let $P_n^*(\boldsymbol{z}|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n^*)$ denote the cumulative distribution function (CDF) corresponding to $p_n^*(\boldsymbol{z}|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n^*)$. 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 $\lim_{\epsilon \downarrow 0} \pi_\epsilon(\boldsymbol{z}|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n^*)$ 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 $\pi_0^{\otimes}(\boldsymbol{z})$ on $\mathbb{R}^n$. Specifically,

(a) *($\Pi_n(\mathcal{Z}_n) > 0$)*: In this case, $p_n^*(\boldsymbol{z}|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n^*)$ is given by

$$\frac{dP_n^*(\boldsymbol{z}|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n^*)}{d\boldsymbol{z}} = \frac{1}{\Pi_n(\mathcal{Z}_n)} \pi_0^{\otimes n}(\boldsymbol{z}), \quad \boldsymbol{z} \in \mathcal{Z}_n, \tag{C.14}$$

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 $P_n^*(\boldsymbol{z}|\boldsymbol{Y}_n, \boldsymbol{X}_n, \boldsymbol{w}_n^*)$ concentrates on the highest dimensional manifold, denoted by $\mathcal{Z}_n^{(p)}$, and is given by

$$\frac{dP_n^*(\boldsymbol{z}|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n^*)}{d\nu}(\boldsymbol{z}) = \frac{\pi_0^{\otimes n}(\boldsymbol{z}) \left(\det(\nabla_{\boldsymbol{t}}^2 U_n(\boldsymbol{z}))\right)^{-1/2}}{\int_{\mathcal{Z}_n^{(p)}} \pi_0^{\otimes n}(\boldsymbol{z}) \left(\det(\nabla_{\boldsymbol{t}}^2 U_n(\boldsymbol{z}))\right)^{-1/2} d\nu}, \quad \boldsymbol{z} \in \mathcal{Z}_n^{(p)}, \tag{C.15}$$

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

By Theorem 3.2 and Lemma 4.2 in Liang et al. (2025), if the target model is noise-additive and $U_n(\cdot)$ is specified as in (C.4), then $P_n^*(\boldsymbol{z}|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n^*)$ in (C.15) can be reduced to

$$\frac{dP_n^*(\boldsymbol{z}|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n^*)}{d\nu} = \frac{\pi_0^{\otimes n}(\boldsymbol{z})}{\int_{\mathcal{Z}_n} \pi_0^{\otimes n}(\boldsymbol{z}) d\nu}. \tag{C.16}$$

That is, under the consistency of $\boldsymbol{w}_n^*$, $p_n^*(\boldsymbol{z}|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n^*)$ is reduced to a truncated density function of $\pi_0^{\otimes n}(\boldsymbol{z})$ 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 $\boldsymbol{Z}_n$ is asymptotically invariant to the inverse function we learned given its consistency.

C.3 The NE-EFI Method

The EFI algorithm (Algorithm C.1) from the previous section has been extended to various model types, including those for estimating individual treatment effects (Kim & Liang, 2025) and Physics-Informed Neural Networks (PINNs) (Shih et al., 2025). However, these extensions primarily focus on the systematic component of the data-generating equations (i.e., the mean structure). They assume the noise distribution remains known and is i.i.d. Gaussian, denoted as $\pi_0^{\otimes n}(\boldsymbol{z}) = \prod_{i=1}^n \phi(z_i)$, where $\phi(\cdot)$ is the standard Gaussian density function. While this Gaussian assumption underpins much of classical statistical inference and machine learning, real-world data often deviate from this ideal.

Recognizing this limitation, Wang & Liang (2026) proposed Nonparametric-Error EFI (NE-EFI). This approach generalizes the EFI framework to accommodate *unknown and non-Gaussian* noise distributions through a multi-stage process that combines the original EFI with kernel density estimation (KDE) (Silverman, 2018; Parzen, 1962).

Here, we briefly review their approach, beginning with their assumed setting. Assume that the errors $z_i \in \mathbb{R}$ are univariate i.i.d. draws from an *unknown* continuous distribution, and consider additive

noise models *without an intercept*. For $i = 1, 2, \ldots, n$, this is defined as:

$$y_i = \underbrace{f(\boldsymbol{x}_i, \boldsymbol{\theta})}_{\text{No intercept}} + z_i, \quad z_i \overset{i.i.d.}{\sim} \underbrace{\pi_0(z)}_{\text{Unknown}}. \tag{C.17}$$

Since EFI is equation-solving in nature: as $\epsilon \downarrow 0$, the Boltzmann factor $e^{-U_n/\epsilon}$ concentrates on the zero-energy set $\mathcal{Z}_n$. Consequently, the imputed errors are driven toward values that satisfy the data-generating equations in (C.17). As a result, *pointwise* error recovery remains relatively insensitive to the initial working error model, provided the sampler adequately explores the relevant zero-energy region. However, the induced fiducial density for $\boldsymbol{Z}_n$ explicitly depends on the baseline density used during sampling. If the working density $\pi_0(\cdot)$ is misspecified, the resulting fiducial distribution—and consequently, the uncertainty quantification—can be significantly distorted. Specifically, under the additive noise setting in (C.17), the EFI of $\boldsymbol{Z}_n$ manifests as a truncated distribution of $\pi_0^{\otimes n}(\boldsymbol{z})$ on $\mathcal{Z}_n$, as given in (C.16).

To mitigate this issue, they utilize the imputed errors to reconstruct the true error distribution via KDE, ensuring the correctness of the resulting EFD (as illustrated in Figure C.3). Let $Z_{i,t}^{(k)}$ denote the imputed errors at MCMC iteration k during run t of NE-EFI. The indices range over $i = 1, \ldots, n$, $k = 1, \ldots, \mathcal{K} + M$, and $t = 1, \ldots, T$. Here, T denotes the terminal run at which NE-EFI converges, and $\mathcal{K} + M$ indicates the final MCMC iteration, comprising M samples collected post burn-in. Accordingly, we denote the full collection of errors at the t-th run as $\boldsymbol{Z}_{n,t}^{(k)} = \{z_{1,t}^{(k)}, \ldots, z_{n,t}^{(k)}\}$.

Under regularity conditions regarding the geometry of the tubular neighborhood of $\mathcal{Z}_n$, they show that the NE-EFI operator is stable as $\epsilon \to 0$. Specifically, it acts as an approximate contractive operator (with respect to the 1-Wasserstein distance) up to a bias floor that vanishes as $\epsilon \to 0$ (Wang & Liang, 2026, Theorem 3.1). Furthermore, assuming suitable conditions for the KDE bandwidth $h > 0$, taking $\epsilon \to 0$ and that the $\boldsymbol{\theta}$ in (C.17) is *identifiable* after z_i's are known, they proved that at the terminal run of NE-EFI,

$$\Pi_n^{(T)}(x) \to \Pi_0(x)$$

uniformly for all $x \in \mathbb{R}$ as $n \to \infty$, where $\Pi_n^{(T)}(x)$ and $\Pi_0(x)$ are the cumulative distribution functions (CDFs) of $\pi_{n,h_n}^{(T)}$ and π_0, respectively (Wang & Liang, 2026, Theorem 3.3).

Remark C.1 (Handling Intercepts and Zero-Mean KDE). For models incorporating an intercept, Wang & Liang (2026) suggests an asymptotic normality-based treatment (see their Remark F.1). Notably, they apply a location-scale transformation after the second run of NE-EFI, absorbing the location and scale parameters into the imputed errors.

In the present work, however, DNN models lack an explicit intercept term. To ensure the resulting KDE maintains a mean of zero, we establish $\pi_{\epsilon,n,h_n}^{(t)}(\boldsymbol{Z}_n | \boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n)$ in (C.20) by computing a centered KDE:

$$\pi_{n,h_n}^{(t)}(z) = \frac{1}{nh_n} \sum_{i=1}^{n} K\left(\frac{z - \tilde{Z}_{i,t-1}^{(\mathcal{K}+M)}}{h_n}\right), \tag{C.18}$$

where $\tilde{Z}_{i,t-1}^{(\mathcal{K}+M)} = Z_{i,t-1}^{(\mathcal{K}+M)} - \bar{Z}_{n,t-1}^{(\mathcal{K}+M)}$ and $\bar{Z}_{n,t-1}^{(\mathcal{K}+M)} = \frac{1}{n}\sum_{i=1}^{n} Z_{i,t-1}^{(\mathcal{K}+M)}$.

Remark C.2 (Gaussian Prediction Interval). Algorithm C.3 outlines the procedure for forming prediction intervals using NE-EFI when π_0 is unknown and potentially non-Gaussian. It is worth noting that if π_0 is known and assumed to be Gaussian, one can still form prediction intervals following Algorithm C.3 with minor modifications: execute the vanilla EFI algorithm (Algorithm C.1) instead of Algorithm C.2 in step (i), and replace step (iii) by drawing $z'_k \sim \mathcal{N}(0, \hat{\sigma}_k^2)$, where $\hat{\sigma}_k$ is the EFI estimate of σ at MCMC iteration k.

C.4 Diagnosis

One of the advantages of EFI and NE-EFI is that they are naturally equipped with several diagnostic tools, making EFI-based approaches reliable and verifiable. In standard EFI where the error distribution π_0 is known, Liang et al. (2025) and Shih et al. (2025) suggest diagnosing convergence by analyzing QQ-plots of the imputed errors and verifying that the energy function U_n minimizes to zero.

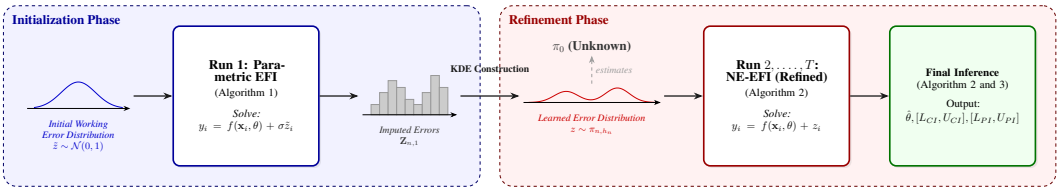


Figure C.3: Schematic of the NE-EFI framework.

Algorithm C.2: Nonparametric Error EFI (NE-EFI)

(i) (Initial Run) Execute Algorithm C.1 with a small temperature parameter $\epsilon > 0$ under a Gaussian working error model: $y_i = f(\boldsymbol{x}_i, \boldsymbol{\theta}) + \sigma\tilde{z}_i$, where $\tilde{z}_i \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$ and σ is treated as unknown.

(ii) (Error Extraction) Extract the imputed errors $\boldsymbol{Z}_{n,1}^{(\mathcal{K}+M)} = \hat{\sigma}\tilde{Z}_i^{(\mathcal{K}+M)}$, and construct the kernel density estimator $\pi_{n,h_n}^{(1)}(z)$ using $\boldsymbol{Z}_{n,1}^{(\mathcal{K}+M)}$:

$$\pi_{n,h_n}^{(1)}(z) = \frac{1}{nh_n}\sum_{i=1}^{n} K\left(\frac{z - Z_{i,1}^{(\mathcal{K}+M)}}{h_n}\right), \tag{C.19}$$

where the bandwidth h_n is a pre-specified bandwidth. Set a convergence threshold $\delta > 0$, $W_1 = 2\delta$, and the iteration counter $t = 2$.

while $W_1 > \delta$ **do**

(iii) (Refined Run) Re-run Algorithm C.1 with target distribution π_ϵ in the SGLD update (C.8) replaced by:

$$\pi_{\epsilon,n,h_n}^{(t)}(\boldsymbol{Z}_n|\boldsymbol{X}_n, \boldsymbol{Y}_n, \boldsymbol{w}_n) \propto \pi_{n,h_n,t}^{\otimes n}(z_i)e^{-U_n(\boldsymbol{Y}_n, \boldsymbol{X}_n, \boldsymbol{Z}_n, \boldsymbol{w}_n)/\epsilon} \tag{C.20}$$

where $\pi_{n,h_n,t}^{\otimes n}(z_i) = \prod_{i=1}^{n}\pi_{n,h_n}^{(t-1)}(z_i)$.

(iv) (KDE Construction) Construct the kernel density estimator $\pi_{n,h_n}^{(t)}(z)$ using $\boldsymbol{Z}_{n,t}^{(\mathcal{K}+M)}$, defined as:

$$\pi_{n,h_n}^{(t)}(z) = \frac{1}{nh_n}\sum_{i=1}^{n} K\left(\frac{z - Z_{i,t}^{(\mathcal{K}+M)}}{h_n}\right), \tag{C.21}$$

where the bandwidth h_n is a pre-specified bandwidth.

(v) Convergence assessment Calculate

$$W_1 = \mathbb{W}_1(\pi_{n,h_n}^{(t)}, \pi_{n,h_n}^{(t-1)}).$$

(vi) (Update Iteration) Increment $t \leftarrow t + 1$.

end

(vi) (Statistical Inference) Set $T = t - 1$. Perform statistical inference using the fiducial samples collected from run T, as detailed in (C.13).

When π_0 is unknown and non-Gaussian, however, the QQ-plot can no longer serve as a confirmation of ground truth. Instead, it functions as an exploratory tool: if the initial run exhibits significant deviation from theoretical quantiles of the presumed π_0, this hints the necessary refinement runs provided by NE-EFI.

Beyond exploratory QQ-plots, convergence of NE-EFI is assessed using standard MCMC diagnostics (e.g., trace plots, autocorrelation, effective sample size, and Gelman–Rubin statistics (Gelman & Rubin, 1992)), together with two method-specific checks: (i) monitoring the trajectory of the energy function U_n, which should converge to (near) zero, and (ii) comparing imputed latent variables across consecutive runs, $\boldsymbol{Z}_{n,t}^{(\mathcal{K}+M)}$ versus $\boldsymbol{Z}_{n,t+1}^{(\mathcal{K}+M)}$, to empirically assess the convergence condition of Algorithm C.2.

Algorithm C.3: Prediction Interval Construction for y_{n+1}

(i) (Input) Imputed errors $\{z_{i,T}^{(k)}\}_{i=1}^{n}$, parameter samples $\{\bar{\boldsymbol{\theta}}_{n,T}^{(k)}\}_{k=\mathcal{K}+1}^{\mathcal{K}+M}$ (run T of NE-EFI in Algorithm C.2), test point $\boldsymbol{x}_{n+1}$, bandwidth h_{pred}.
(ii) (Fiducial Error Collection) for $k = \mathcal{K}+1, \ldots, \mathcal{K}+M$ **do**
  (iii) Error Sampling: Draw z'_k from $\{z_{i,T}^{(k)}\}_{i=1}^{n}$ uniformly.
  (iv) Smoothing: Generate a jitter term $\delta_k \sim \mathcal{N}(0, h_{\text{pred}}^2)$.
  (v) Prediction: Compute the response value as $y_{n+1}^{(k)} = f(\boldsymbol{x}_{n+1}, \bar{\boldsymbol{\theta}}_{n,T}^{(k)}) + z'_k + \delta_k$.
end
(vi) (Interval Construction) Collect the ensemble of predicted values $\mathcal{Y} = \{y_{n+1}^{(k)}\}_{k=\mathcal{K}+1}^{\mathcal{K}+M}$. Compute the empirical quantiles $[q_{\alpha/2}, q_{1-\alpha/2}]$ of $\mathcal{Y}$ to obtain the $(1-\alpha)$ prediction interval.

D Detailed Simulation Studies for Section 5

This section provides the full experimental results outlined in Section 5. To ensure a self-contained and smooth presentation, some content here overlaps with Section 5. The descriptions and evaluation criteria detailed below also apply to the supplementary experiments in Appendix F.

Goal and Evaluation. In scientific and structured regression problems, the goal is twofold: predicting future noisy observations Y^* and quantifying uncertainty about the latent function $f^\star$ representing the true underlying signal. Prediction intervals (PIs) for $Y^*(x)$ assess predictive validity, whereas confidence intervals (CIs) for $f^\star(x)$ assess our uncertainty regarding the latent scientific target. Therefore, we report both the CI coverage for $f^\star(x)$ and the PI coverage for $Y^*(x)$, along with their corresponding interval widths. Additionally, the mean squared errors (MSEs) for estimating $f^\star(x)$ are reported to evaluate estimation accuracy.

Coverage is evaluated in the frequentist sense across 100 independently simulated datasets. CI coverage is defined as the frequency with which the constructed intervals contain the true latent function values $f^\star(x)$. Similarly, PI coverage is the frequency with which the intervals contain independently generated future responses Y^*. A nominal interval is considered well-calibrated if its empirical coverage over repeated simulations closely matches its target rate. Following the convention in the conformal prediction literature, our target coverage rates are 95% for CIs and 90% for PIs. All metrics are computed on an independent evaluation set: for 1-dimensional cases, 100 evaluation points x are equally spaced in $\Omega \subset \mathbb{R}^1$; for 5-dimensional examples, 5,000 evaluation points $\boldsymbol{x}$ are drawn uniformly from $\Omega \subset \mathbb{R}^5$. $f^\star(x)$ and Y^* for evaluation can be generated accordingly. Different sizes n are used in different examples.

The simulations focus on three primary challenges: (i) **Constraint flexibility:** While existing nonparametric regression methods provide useful uncertainty quantification under standard smoothness assumptions, they are limited in the types of structural or physical constraints they can naturally accommodate (e.g., the 1-D Poisson equation). (ii) **Latent-function uncertainty:** PINN-style penalty methods incorporate constraints via soft penalties, which can be effective for point estimation. However, heuristic uncertainty methods like Deep Ensembles or Monte Carlo Dropout do not directly provide calibrated uncertainty quantification for the latent function $f^\star$. (iii) **Noise robustness:** Existing interval methods are often sensitive to non-Gaussian or nonstandard noise distributions (e.g., highly skewed distributions or a bimodal Gaussian mixture with well-separated modes). Scale parameter σ is considered unknown in all cases.

Crucially, we demonstrate that the proposed EFI framework can perform valid statistical inference for dense neural networks when an embedded low-dimensional structure is enforced. This structural constraint makes the model identifiable, thereby enabling robust inference even when the true error distribution is unknown.

Implementation. For confidence intervals and point estimation, we compare EFI against neural network baselines—PINNs with Monte Carlo dropout (Zhang et al., 2019; Gal & Ghahramani, 2016), Deep Ensembles (Lakshminarayanan et al., 2016), and Bayesian PINNs (Yang et al., 2021)—as well as classical nonparametric smoothing methods. The classical baselines include cubic regression

splines (via the *gam* function in the R package *mgcv*, Wood & Wood, 2015), cubic local polynomials (via the R package *locfit*, Loader & Liaw, 2025), and Smoothing Spline ANOVA (SS-ANOVA, via the *ssanova* function in the R package *gss*, Gu, 2014). Smoothing parameters were selected using restricted maximum likelihood (REML) for *gam* and generalized cross-validation (GCV) for *ssanova*. For non-Gaussian noise settings where refined runs of EFI are needed, we set the bandwidth $h = 0.3$, which corresponds to a slightly over-smoothing option as suggested by Wang & Liang (2026), and $h_{\text{pred}} = 0$. See Algorithms 1, C.2, and C.3 for details and Appendix G for full hyperparameter settings.

Prediction uncertainty for the competing methods was quantified using split conformal prediction (SCP), with the n observations evenly divided between model fitting and calibration. This follows the default implementation in the R package *conformalInference* (Lei et al., 2018). Prediction intervals for MC dropout and deep ensembles were constructed according to Algorithms G.1 and G.2, respectively. In contrast, EFI directly produces prediction intervals, as described in Algorithm C.3 and Remark C.2, together with confidence intervals, thereby providing a unified framework for uncertainty quantification of both the latent function and future observations.

Results are aggregated over 100 independent trials and reported as the mean with standard error of the mean (SE) in parentheses. In what follows, we present three distinct examples: (i) a problem with a differentiable constraint, (ii) a problem with a non-differentiable constraint, and (iii) a high-dimensional problem featuring a three-way interaction.

D.1 1-D Poisson Equation with Bimodal Mixture Gaussian Errors

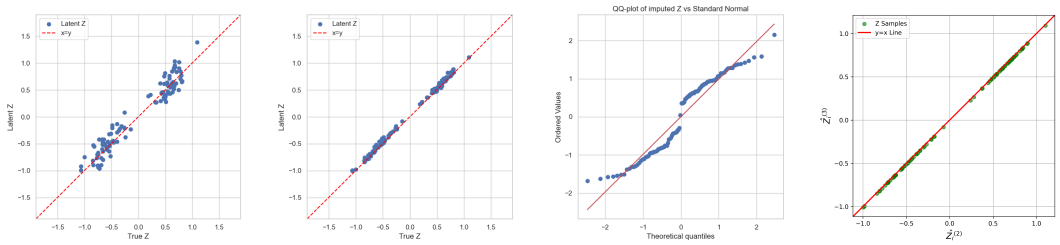
We consider a 1D Poisson equation following Shih et al. (2025) and Yang et al. (2021), with true solution $f^\star(x) = \sin^3(6x)$ on $x \in [-0.7, 0.7]$. The system is governed by a deterministic PDE constraint enforced at $m = 200$ collocation points:

$$0.01\, f''(x) = g(x), \qquad g(x) = 1.08 \sin(6x)\big(2\cos^2(6x) - \sin^2(6x)\big). \tag{D.1}$$

Observations follow $y_i = f^\star(x_i) + z_i$, where $z_i \overset{\text{i.i.d.}}{\sim} 0.5\,\mathcal{N}(-0.6, 0.2^2) + 0.5\,\mathcal{N}(0.6, 0.2^2)$ for $i = 1, \ldots, n$ with $n = 100$. The present PDE constraint is out of the scope of classical nonparametric methods. Standard Bayesian PINNs struggle here due to the strictly deterministic PDE constraint and heavily non-Gaussian (bimodal) noise. We thus compare our EFI framework against standard PINNs (with 1% and 5% MC dropout), deep ensembles, and SCP. To assess sensitivity to the penalty parameter in PINNs, we vary $\lambda_{\text{str}} \in \{1, 10, 100\}$. In EFI, η_{str} weights the constraints in equation (7) simply to guide early training. Because the energy $U \to 0$ as temperature $\epsilon \to 0$, the constraints are eventually *strictly satisfied* regardless of the initial η_{str}. We verify this robustness using dynamic η_{str} schedules (see Appendix G).

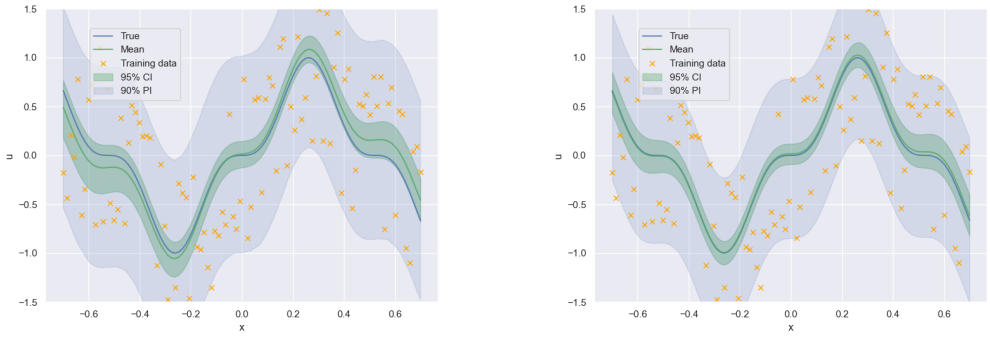
As shown in Table D.1, baseline PINNs fail to provide robust inference. They are highly sensitive to dropout rates and λ_{str}, with CI coverages swinging from 66.9% to 99.9%. Their higher MSE likely stems from the standard PINN loss implicitly assuming Gaussian noise (3) can be seen as a Gaussian likelihood with a prior governed by $\lambda_{\text{str}}\, \mathcal{R}_{\text{str}}(f_{\boldsymbol{\theta}})$). Among the reported dropout settings, λ_{str}=1 with dropout 5% gives CI coverage closest to the nominal 95% level (92.78%), with a width of 0.5193, approximately 1.8–1.9 times the EFI widths (0.2773–0.2925). Deep ensembles achieve competitive MSE but suffer severe CI collapse: the strict deterministic PDE restricts the hypothesis space, causing ensemble members to lose functional diversity and converge to identical solutions, leading to catastrophic CI coverage ($< 1\%$). Adding subagging improves coverage to only $\sim 62\%$. The Gaussian-working-model PIs from dropout and ensembles, including subagging, cover at 93.88–97.59% and have widths of 1.9870–2.1170, which are longer than those produced by EFI. SCP and EFI attain coverage close to the nominal 90% level (90.27% and 89.51–89.57%, respectively), with narrower PIs; in particular, EFI widths are 1.7229–1.7269, compared with 1.8527 for SCP, demonstrating its superior inference efficiency. Additionally, EFI achieves consistently lower MSE and near-nominal CI/PI coverage across all η_{str} schedules.

Finally, Figure D.1 presents latent variable imputations, a QQ plot justifying the need for a refined run, and convergence diagnostics. As shown in Figure D.2, this refined EFI run successfully shrinks the uncertainty intervals, properly rectifying the initial Gaussian-based approximations.



(a) EFI Run 1 (b) EFI Run 2 (c) QQ plot (d) $\boldsymbol{Z}^{(2)}$ vs. $\boldsymbol{Z}^{(3)}$

Figure D.1: Latent variable imputation, QQ plot, and convergence diagnostics for the 1D-Poisson equation case. Note that (a) Run 1: $\boldsymbol{Z}$ vs. $\hat{\sigma}\tilde{\boldsymbol{Z}}^{(1)}$ and (b) Run 2: $\boldsymbol{Z}$ vs. $\boldsymbol{Z}^{(2)}$.



(a) EFI Run 1 (b) EFI Run 2 (with KDE refinement)

Figure D.2: 95% CI and 90% PI for the 1-D Poisson example.

Table D.1: Results on 1-D Poisson example with bimodal mixture Gaussian errors. $\eta_{\text{str}}=1 \to 5$ at 10% means η_{str} increases from 1 to 5 at the first 10% of training. See Appendix G.

Method	MSE	CI Coverage (95%)	CI Range	PI Coverage (90%)	PI Range
EFI ($\eta_{\text{str}}=1$)	0.0045 (0.0006)	0.9545 (0.0153)	0.2925 (0.0035)	0.8957 (0.0040)	1.7269 (0.0059)
EFI ($\eta_{\text{str}}=1 \to 5$ at 10%)	0.0042 (0.0005)	0.9461 (0.0163)	0.2773 (0.0023)	0.8957 (0.0039)	1.7229 (0.0058)
EFI ($\eta_{\text{str}}=1 \to 10$ at 10%)	0.0043 (0.0006)	0.9463 (0.0166)	0.2791 (0.0033)	0.8951 (0.0038)	1.7232 (0.0059)
PINN ($\lambda_{\text{str}}=1$, dropout 1%)	0.0199 (0.0011)	0.6688 (0.0180)	0.2690 (0.0041)	0.9388 (0.0032)	1.9870 (0.0069)
PINN ($\lambda_{\text{str}}=1$, dropout 5%)	0.0197 (0.0011)	0.9278 (0.0082)	0.5193 (0.0053)	0.9467 (0.0031)	2.0255 (0.0068)
PINN ($\lambda_{\text{str}}=10$, dropout 1%)	0.0126 (0.0009)	0.8061 (0.0170)	0.2855 (0.0015)	0.9568 (0.0026)	2.0221 (0.0066)
PINN ($\lambda_{\text{str}}=10$, dropout 5%)	0.0129 (0.0009)	0.9884 (0.0033)	0.6021 (0.0034)	0.9653 (0.0023)	2.0708 (0.0063)
PINN ($\lambda_{\text{str}}=100$, dropout 1%)	0.0092 (0.0009)	0.8976 (0.0159)	0.3168 (0.0019)	0.9656 (0.0022)	2.0428 (0.0064)
PINN ($\lambda_{\text{str}}=100$, dropout 5%)	0.0094 (0.0009)	0.9993 (0.0007)	0.7361 (0.0034)	0.9759 (0.0019)	2.1170 (0.0063)
PINN ($\lambda_{\text{str}}=100$) + SCP	0.0205 (0.0017)	–	–	0.9027 (0.0048)	1.8527 (0.0143)
Deep Ensemble ($\lambda_{\text{str}}=10$, $M=50$)	0.0125 (0.0009)	0.0096 (0.0019)	0.0025 (0.0001)	0.9575 (0.0029)	2.0143 (0.0065)
Deep Ensemble ($\lambda_{\text{str}}=100$, $M=50$)	0.0091 (0.0009)	0.0039 (0.0006)	0.0009 (0.0000)	0.9654 (0.0025)	2.0323 (0.0064)
Deep Ensemble ($\lambda_{\text{str}}=1000$, $M=50$)	0.0084 (0.0009)	0.0038 (0.0006)	0.0008 (0.0000)	0.9681 (0.0023)	2.0403 (0.0064)
Deep Ensemble + Subagging ($\lambda_{\text{str}}=10$, $M=50$, sub=0.8)	0.0126 (0.0009)	0.6175 (0.0236)	0.1975 (0.0013)	0.9567 (0.0029)	2.0106 (0.0066)
Deep Ensemble + Subagging ($\lambda_{\text{str}}=100$, $M=50$, sub=0.8)	0.0092 (0.0009)	0.6279 (0.0286)	0.1662 (0.0013)	0.9650 (0.0025)	2.0300 (0.0064)
Deep Ensemble + Subagging ($\lambda_{\text{str}}=1000$, $M=50$, sub=0.8)	0.0084 (0.0009)	0.6241 (0.0312)	0.1584 (0.0014)	0.9674 (0.0025)	2.0394 (0.0064)

D.2 1-D Poisson Equation with Gaussian Errors

We evaluate EFI and baseline methods using the 1D Poisson setup detailed in Appendix D.1, but here we introduce standard Gaussian errors $z_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.1^2)$ for $i = 1, \ldots, 100$. Under these conditions, the likelihood of standard BPINNs is correctly specified, providing a strong baseline for comparison alongside deep ensembles.

As shown in Table D.2, all methods, except for BPINN with $\sigma_{\text{diff}}=0.1$, produce comparable MSEs for the point estimates. However, their uncertainty quantification varies drastically. Consistent with the non-Gaussian example, deep ensembles exhibit severe CI undercoverage, yielding near-zero CI widths and CI coverage of 3.11–5.50%, far below the nominal 95% level. Their Gaussian-working-model PIs have coverage of 88.91–89.53%, close to the nominal 90% level. BPINNs successfully hit the target 95% CI and 90% PI coverages for larger values of the noise for PDE observation σ_{diff} ($\sigma_{\text{diff}} \in \{0.1, 0.01\}$). However, when σ_{diff} is reduced to 0.001 to approximate the strictly deterministic PDE constraint of this problem, BPINNs catastrophically fail, dropping to 0.0% CI coverage. This failure in the nearly deterministic limit is a known structural limitation of the BPINN formulation (5), as pointed out by Shih et al. (2025).

In contrast, our proposed EFI framework attains the lowest overall MSE (0.0002), valid PI coverage ($\sim 89.4\%$), and valid CI coverage ($\sim 94.5\%$). It is crucial to highlight a major practical advantage of EFI in this comparison: BPINNs assume that the true noise scale σ is *known* and provided to the model, whereas the EFI results presented here assume σ is *unknown*. Despite this informational disadvantage, EFI accurately estimates the latent noise scale directly from the data, yielding $\hat{\sigma}_{\text{EFI}} = 0.0969$ (standard error 0.0007), which closely recovers the true $\sigma = 0.1$.

Table D.2: Results on 1-D Poisson example with Gaussian errors. †: BPINNs assume σ is known; while EFI present here assumes σ is unknown.

Method	MSE	CI Coverage (95%)	CI Range	PI Coverage (90%)	PI Range
† **EFI** ($\eta_{\text{str}} = 0.25$)	0.0003 (0.0000)	0.9456 (0.0116)	0.0686 (0.0006)	0.8935 (0.0040)	0.3257 (0.0024)
EFI ($\eta_{\text{str}} = 0.5$)	0.0003 (0.0000)	0.9470 (0.0113)	0.0668 (0.0005)	0.8936 (0.0040)	0.3240 (0.0024)
EFI ($\eta_{\text{str}} = 1.0$)	0.0002 (0.0000)	0.9411 (0.0121)	0.0631 (0.0005)	0.8948 (0.0039)	0.3235 (0.0024)
Deep Ensemble ($\lambda_{\text{str}}=10$, $M=50$)	0.0003 (0.0000)	0.0550 (0.0046)	0.0023 (0.0001)	0.8891 (0.0039)	0.3207 (0.0022)
Deep Ensemble ($\lambda_{\text{str}}=100$, $M=50$)	0.0002 (0.0000)	0.0311 (0.0059)	0.0009 (0.0000)	0.8943 (0.0037)	0.3230 (0.0023)
Deep Ensemble ($\lambda_{\text{str}}=1000$, $M=50$)	0.0002 (0.0000)	0.0316 (0.0052)	0.0008 (0.0000)	0.8953 (0.0037)	0.3242 (0.0023)
BPINN ($\sigma_{\text{diff}}=0.1$)	0.1552 (0.1548)	0.9765 (0.0055)	0.1615 (0.0742)	0.9069 (0.0032)	0.4107 (0.0740)
† BPINN ($\sigma_{\text{diff}}=0.01$)	0.0002 (0.0000)	0.9636 (0.0106)	0.0583 (0.0003)	0.9020 (0.0031)	0.3321 (0.0001)
BPINN ($\sigma_{\text{diff}}=0.001$)	0.0006 (0.0001)	0.0000 (0.0000)	0.0000 (0.0000)	0.8936 (0.0033)	0.3284 (0.0001)

Table D.3: Run time comparison for the 1-D Poisson example with Gaussian errors. Training time per epoch measured on an Apple M4 Pro (12-core CPU, 8P+4E). Deep Ensemble runtime $=$ PINN runtime $\times\, M$.

Method	**Time (s/epoch)**
EFI (Ours)	0.0021
PINN	0.0008
BPINN	0.0110
Deep Ensemble	$0.0008 \times M$

D.3 Periodic Function

We evaluate a non-smooth periodic regression problem with the true solution $f^\star(x) = |\sin(2\pi x)| + 0.3\sin(4\pi x)$ on $x \in [-0.5, 0.5]$, given $n = 300$ observations. The structural constraint is a periodicity of $T = 0.5$:

$$C^{(1)}(f;\, \tilde{x}) = f(\tilde{x}) - f(\tilde{x} + T) = 0, \tag{D.2}$$

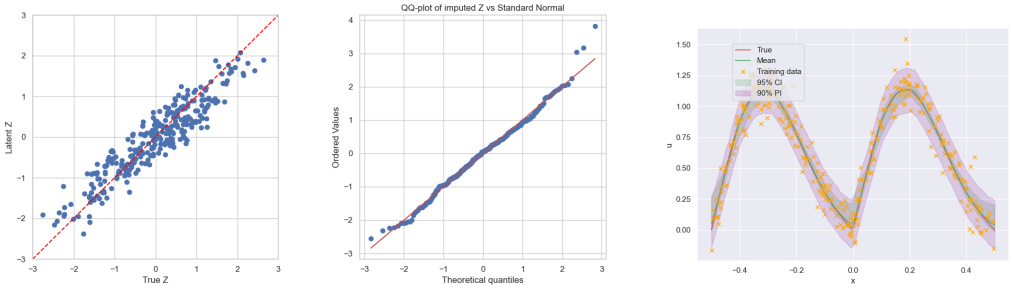
enforced at $m = 200$ collocation points on $[-0.5, 0]$. This example demonstrates our framework's flexibility in handling non-differentiable constraints. We evaluate two noise settings: standard Gaussian errors $z_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.1^2)$ and the bimodal non-Gaussian mixture utilized in the 1D Poisson setting (see Appendix F for skewed errors).

Table D.4 reports the Gaussian noise results. While deep learning baselines (PINNs, deep ensembles, and BPINNs) can achieve competitive MSEs, their performance remains highly sensitive to the data-constraint balancing parameter λ_{str}. Crucially, none of these DNN-based methods reach the 95% nominal CI coverage, peaking at only $\sim 92.6\%$ even in the best-case scenario. Conversely, classical nonparametric methods (Cubic Smoothing Splines and Local Polynomials) yield reliable CIs but suffer from higher MSEs than EFI. For PIs, BPINNs consistently under-cover, whereas MC dropout, deep ensemble, SCP and EFI reliably hit the 90% target. Notably, EFI yields a PI with lower variance, which we attribute to its full data utilization compared to SCP's data-splitting approach.

Under bimodal non-Gaussian errors (Table D.5), standard PINN baselines completely fail to adapt, yielding severe under-coverage and higher MSEs. While traditional nonparametric regressions successfully maintain the nominal CI coverage, their implicit Gaussian assumption forces them to produce highly inflated interval widths (more than double the width of EFI). Similarly, while SCP theoretically guarantees the 90% PI coverage, it does so at the cost of wider intervals. MC dropout again tends to produce over-coverage PIs, similar to Table D.1. In contrast, EFI *automatically* adapts to the non-Gaussian error structure without explicit modeling. Consequently, EFI achieves the lowest MSE and produces the most efficient (narrowest) valid CIs and PIs. Refer to Figure D.4 for EFI diagnostics, and Figures D.5 and D.6 for visual comparisons of the learned intervals.

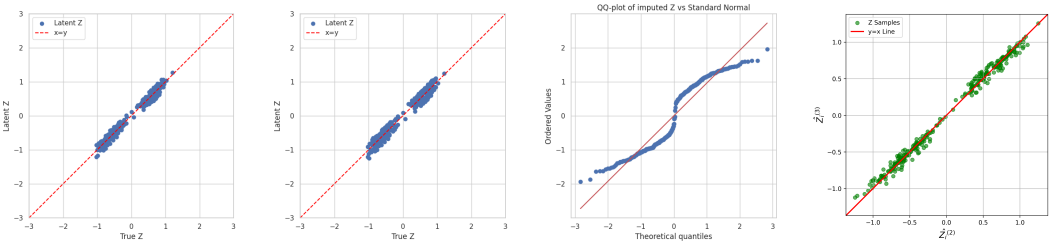
Table D.4: Results on periodic model with Gaussian noise. For BPINN, arrows denote linear annealing of σ_{diff} from 0.1 to the indicated endpoint over the first 40% of training, followed by a constant value for the remaining iterations.

Algorithm	MSE	CI Coverage (95%)	CI Width	PI Coverage (90%)	PI Width
EFI (Ours)	0.0005 (0.0000)	0.9412 (0.0006)	0.1221 (0.0007)	0.8926 (0.0004)	0.3492 (0.0007)
Cubic Smoothing Splines	0.0010 (0.0001)	0.9450 (0.0044)	0.1141 (0.0033)	0.8973 (0.0038)	0.3519 (0.0028)
Cubic Local Polynomial	0.0013 (0.0000)	0.9478 (0.0033)	0.1322 (0.0006)	0.8955 (0.0039)	0.3757 (0.0031)
PINN (λ_{str}=1, dropout 1%)	0.0005 (0.0000)	0.7806 (0.0102)	0.0641 (0.0004)	0.8718 (0.0041)	0.3152 (0.0013)
PINN (λ_{str}=1, dropout 5%)	0.0007 (0.0000)	0.9261 (0.0054)	0.1224 (0.0009)	0.8870 (0.0039)	0.3313 (0.0012)
PINN (λ_{str}=10, dropout 1%)	0.0006 (0.0000)	0.5608 (0.0121)	0.0495 (0.0004)	0.8806 (0.0038)	0.3237 (0.0013)
PINN (λ_{str}=10, dropout 5%)	0.0039 (0.0001)	0.4580 (0.0088)	0.0975 (0.0005)	0.8925 (0.0034)	0.3791 (0.0014)
PINN (λ_{str}=100, dropout 1%)	0.0087 (0.0002)	0.0986 (0.0043)	0.0360 (0.0003)	0.8900 (0.0031)	0.4356 (0.0025)
PINN (λ_{str}=100, dropout 5%)	0.0519 (0.0005)	0.0528 (0.0025)	0.0439 (0.0002)	0.9030 (0.0032)	0.8011 (0.0031)
PINN (λ_{str}=1) + SCP	0.0046 (0.0004)	–	–	0.9007 (0.0040)	0.3951 (0.0040)
Deep Ensemble (λ_{str}=1, M=50)	0.0004 (0.0000)	0.5299 (0.0123)	0.0380 (0.0043)	0.8831 (0.0038)	0.3241 (0.0014)
Deep Ensemble (λ_{str}=10, M=50)	0.0004 (0.0000)	0.6196 (0.0136)	0.0466 (0.0045)	0.8924 (0.0039)	0.3309 (0.0014)
Deep Ensemble (λ_{str}=100, M=50)	0.0006 (0.0000)	0.6443 (0.0111)	0.0575 (0.0025)	0.8965 (0.0038)	0.3395 (0.0013)
Deep Ensemble (λ_{str}=1000, M=50)	0.0030 (0.0001)	0.5142 (0.0082)	0.1091 (0.0018)	0.9148 (0.0029)	0.3939 (0.0015)
BPINN (σ_{diff}=1)	0.0010 (0.0000)	0.7834 (0.0095)	0.0683 (0.0004)	0.8865 (0.0035)	0.3335 (0.0001)
BPINN (σ_{diff}=0.1)	0.0008 (0.0000)	0.7673 (0.0096)	0.0557 (0.0003)	0.8874 (0.0035)	0.3317 (0.0001)
BPINN (σ_{diff} : 0.1 $\rightarrow$ 0.01)	0.0022 (0.0001)	0.4889 (0.0161)	0.0484 (0.0006)	0.8628 (0.0036)	0.3309 (0.0001)
BPINN (σ_{diff} : 0.1 $\rightarrow$ 0.001)	0.0069 (0.0007)	0.0030 (0.0030)	0.0030 (0.0030)	0.8024 (0.0084)	0.3288 (0.0006)



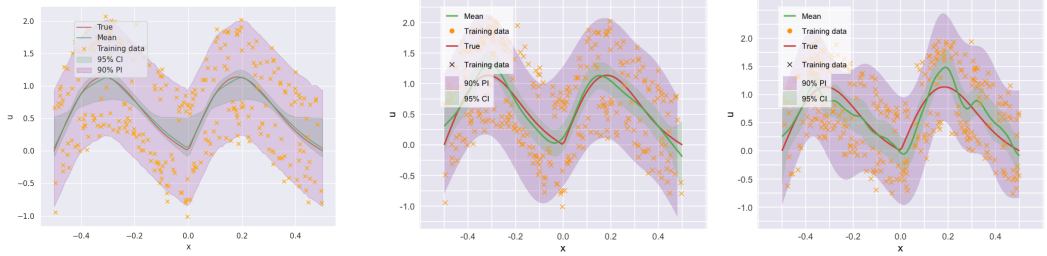
(a) True errors vs imputed errors (b) QQ plot (c) 95% CI and 90% PI

Figure D.3: EFI diagnostic for the Periodic model with Gaussian noise



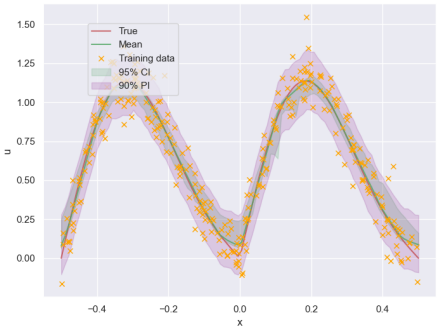
(a) EFI Run 1 (b) EFI Run 2 (c) QQ plot (d) $\boldsymbol{Z}^{(2)}$ vs. $\boldsymbol{Z}^{(3)}$

Figure D.4: Latent variable imputation, QQ plot, and convergence diagnostics for the Periodic model with non-Gaussian noise. Note that (a) Run 1: $\boldsymbol{Z}$ vs. $\hat{\sigma}\tilde{\boldsymbol{Z}}^{(1)}$ and (b) Run 2: $\boldsymbol{Z}$ vs. $\boldsymbol{Z}^{(2)}$.

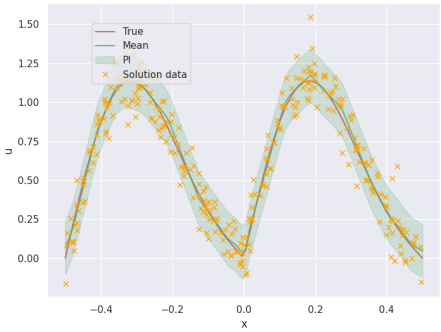


(a) EFI (b) Cubic Smoothing Spline (c) Cubic Local Polynomial

Figure D.5: 95% CIs and 90% PIs for periodic example with non-Gaussian errors



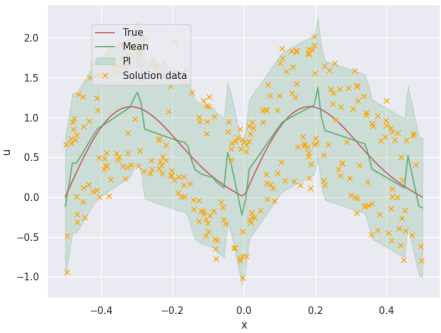
(a) Dropout 95% CI and 90 %PI (Gaussian, λ_{str}=1)



(b) SCP 90% PI (Gaussian, λ_{str}=1)



(c) Dropout 95% CI and 90% PI (Non-Gaussian, λ_{str}=10)



(d) SCP 90% PI (Non-Gaussian, λ_{str}=10)

Figure D.6: Comparison of uncertainty intervals generated by standard PINNs. CIs are estimated using MC dropout (1% rate), while PIs are generated via SCP.

Table D.5: Results on periodic model with bimodal mixture Gaussian errors

Algorithm	MSE	CI Coverage (95%)	CI Width	PI Coverage (90%)	PI Width
EFI (Ours)	0.0041 (0.0003)	0.9402 (0.0100)	0.2972 (0.0070)	0.9103 (0.0030)	1.7617 (0.0041)
Cubic Smoothing Splines	0.0347 (0.0038)	0.9468 (0.0055)	0.6596 (0.0257)	0.8921 (0.0042)	1.8829 (0.0132)
Cubic Local Polynomial	0.0505 (0.0015)	0.9441 (0.0039)	0.8333 (0.0031)	0.8904 (0.0039)	2.2067 (0.0144)
PINN (λ_{str}=1, dropout 1%)	0.0249 (0.0009)	0.5147 (0.0090)	0.1919 (0.0023)	0.9173 (0.0036)	1.9260 (0.0052)
PINN (λ_{str}=1, dropout 5%)	0.0185 (0.0007)	0.6275 (0.0111)	0.2224 (0.0021)	0.9351 (0.0032)	1.9552 (0.0047)
PINN (λ_{str}=10, dropout 1%)	0.0119 (0.0006)	0.2880 (0.0110)	0.0730 (0.0008)	0.9558 (0.0023)	2.0082 (0.0045)
PINN (λ_{str}=10, dropout 5%)	0.0127 (0.0006)	0.3532 (0.0131)	0.1091 (0.0009)	0.9607 (0.0022)	2.0285 (0.0043)
PINN (λ_{str}=100, dropout 1%)	0.0139 (0.0006)	0.1169 (0.0059)	0.0391 (0.0004)	0.9654 (0.0020)	2.0590 (0.0043)
PINN (λ_{str}=100, dropout 5%)	0.0538 (0.0008)	0.0556 (0.0027)	0.0454 (0.0004)	0.9428 (0.0025)	2.1741 (0.0048)
PINN (λ_{str}=10) + SCP	0.0297 (0.0017)	–	–	0.8963 (0.0038)	1.8666 (0.0111)

D.4 5-D NONLINEAR REGRESSION

We evaluate a 5-dimensional regression problem with the true solution $f^\star(\boldsymbol{x}) = 2x_1x_2 + 3x_3^2 + x_1x_2x_4 + 2x_5$ on $\boldsymbol{x} \in [-1, 1]^5$. Observations follow $y_i = f^\star(\boldsymbol{x}_i) + z_i$ with $n = 500$ and $z_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.1^2)$. The structural constraints reflect the polynomial form of $f^\star$: variables x_1, x_2, x_4, and x_5 have linear marginal effects, while x_3 is quadratic. This yields five equality constraints enforced at $m = 200$ collocation points:

$$\frac{\partial^2 f}{\partial x_j^2}(\boldsymbol{x}) = 0 \;\; (j = 1, 2, 4, 5), \qquad \frac{\partial^3 f}{\partial x_3^3}(\boldsymbol{x}) = 0. \tag{D.3}$$

These marginal effects are easily identifiable during exploratory data analysis as shown in Figure D.8 (a). This example tests our method on high-dimensional data with higher-order interactions, challenging classical nonparametric methods like SS-ANOVA (Dai, 2025). For the SS-ANOVA baseline, we consider interactions up to order 3 for x_1, x_2, x_3, x_4, while keeping x_5 linear.

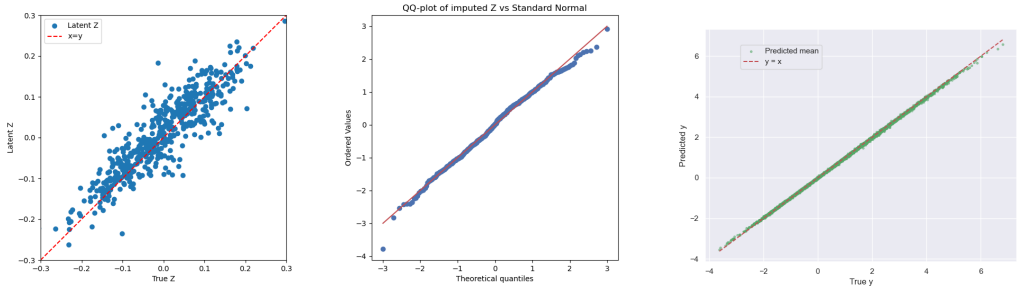
Table D.6 shows that dropout performance is sensitive to the structural-penalty weight and dropout rate. Its MSE ranges from 0.0035 to 0.2773, exceeding EFI's MSE of 0.0014 in every reported setting. Five dropout configurations over-cover in CI (98.40–99.91%), while $\lambda_{\text{str}} = 100$ with dropout 5% under-covers at 90.65%. Deep Ensembles attain lower MSEs than EFI at $\lambda_{\text{str}} = 10$ and 100 (0.0009 and 0.0008), but their MSE rises to 0.0144 at $\lambda_{\text{str}} = 1000$; their CI coverage remains low across all three settings (30.99–35.21%). SS-ANOVA also under-covers at 89.72%, whereas EFI attains CI coverage of 96.17%, close to the nominal 95% level. For PIs, dropout over-covers at 95.63–99.32% and produces wider intervals (0.5864–2.1774) than EFI (0.3371). Ensemble PI coverage is 87.09–89.18%, and EFI attains 88.48%, both below the nominal 90% level.

The slight under-coverage in EFI's PI (88.5% vs. nominal 90%) stems from an underestimation of σ. This overfitting is likely due to the loosely specified constraints; because interactions between variables are not explicitly constrained, the model has excess degrees of freedom. While setting a smaller temperature ϵ theoretically mitigates this by tightening the constraints, the complex gradient dynamics of large models make the strict $\epsilon \to 0$ limit practically challenging to implement.

Visually, EFI-learned surfaces are considerably smoother and generalize better to the ground truth than standard PINNs (compare Figures D.8 and D.9). EFI also achieves lower MSE than all reported dropout configurations. We attribute its improved smoothness to its imputation mechanics: $f^\star(\boldsymbol{x})$ is learned using iteratively imputed latent errors z_i. This effectively acts as dynamic data augmentation, increasing data diversity and forcing the network to learn the underlying structural distribution, whereas baseline DNNs simply overfit to the static training dataset $(\boldsymbol{x}_i, y_i)$.

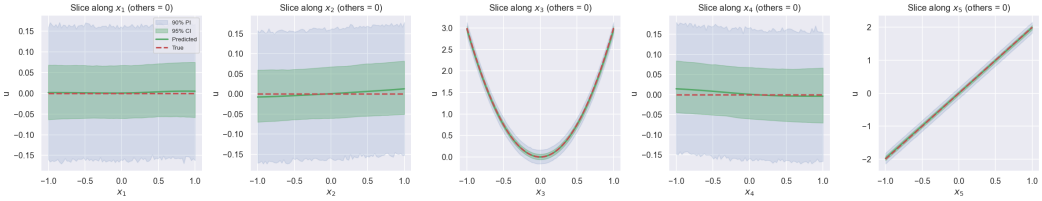
Table D.6: Results on nonlinear-5d model with Gaussian noise.

Algorithm	MSE	CI Coverage (95%)	CI Width	PI Coverage (90%)	PI Width
EFI (Ours)	0.0014 (0.0000)	0.9617 (0.0002)	0.1490 (0.0002)	0.8848 (0.0002)	0.3371 (0.0002)
SS-ANOVA	0.0012 (0.0001)	0.8972 (0.0054)	0.1011 (0.0009)	0.9050 (0.0018)	0.3677 (0.0025)
PINN (λ_{str}=1, dropout 1%)	0.0035 (0.0001)	0.9989 (0.0001)	0.6042 (0.0023)	0.9852 (0.0004)	0.5864 (0.0019)
PINN (λ_{str}=1, dropout 5%)	0.0271 (0.0005)	0.9968 (0.0003)	1.1744 (0.0039)	0.9932 (0.0004)	1.1297 (0.0045)
PINN (λ_{str}=10, dropout 1%)	0.0095 (0.0002)	0.9991 (0.0001)	0.8587 (0.0022)	0.9928 (0.0003)	0.7971 (0.0022)
PINN (λ_{str}=10, dropout 5%)	0.0903 (0.0012)	0.9840 (0.0007)	1.4948 (0.0034)	0.9832 (0.0006)	1.5778 (0.0043)
PINN (λ_{str}=100, dropout 1%)	0.0413 (0.0007)	0.9907 (0.0006)	1.1907 (0.0031)	0.9833 (0.0006)	1.1672 (0.0033)
PINN (λ_{str}=100, dropout 5%)	0.2773 (0.0040)	0.9065 (0.0024)	1.6783 (0.0060)	0.9563 (0.0009)	2.1774 (0.0103)
PINN (λ_{str}=10) + SCP	0.0028 (0.0001)	–	–	0.9049 (0.0019)	0.3772 (0.0023)
Deep Ensemble (λ_{str}=10, M=50)	0.0009 (0.0000)	0.3099 (0.0084)	0.0190 (0.0004)	0.8709 (0.0014)	0.3163 (0.0010)
Deep Ensemble (λ_{str}=100, M=50)	0.0008 (0.0000)	0.3521 (0.0048)	0.0258 (0.0002)	0.8788 (0.0014)	0.3224 (0.0010)
Deep Ensemble (λ_{str}=1000, M=50)	0.0144 (0.0003)	0.3288 (0.0044)	0.0921 (0.0005)	0.8918 (0.0013)	0.4904 (0.0020)

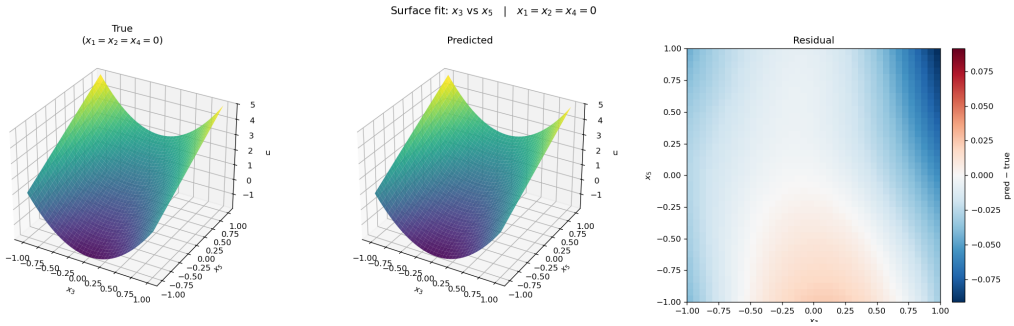


(a) True errors vs imputed errors (b) QQ plot (c) True y vs predicted $\hat{y}$

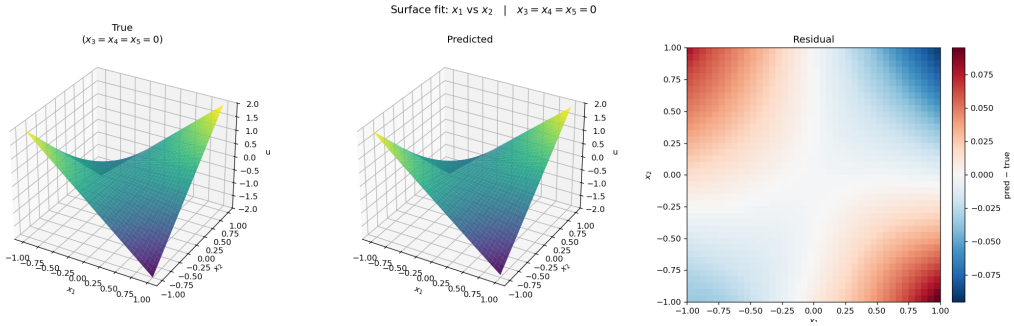
Figure D.7: EFI diagnostic for 5D-nonlinear example



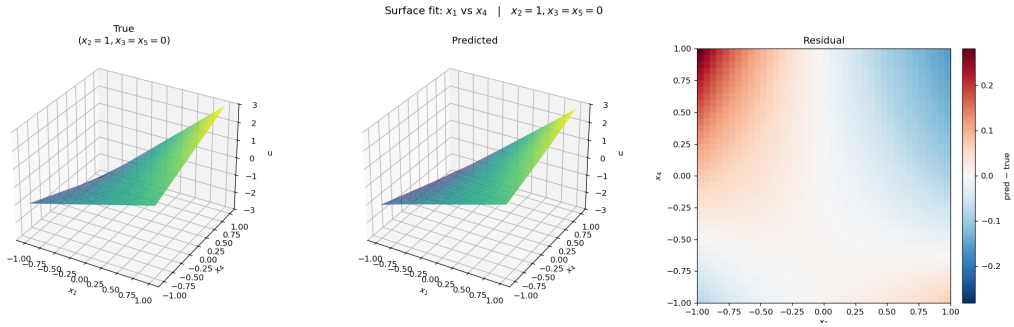
(a) Marginal effects with 95% CIs and 90% PIs



(b) Interaction of x_3 and x_5

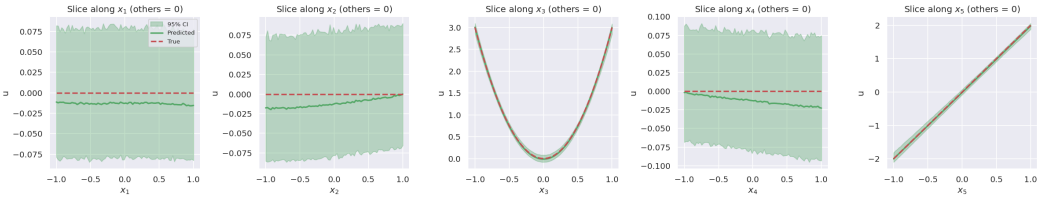


(c) Interaction of x_1 and x_2

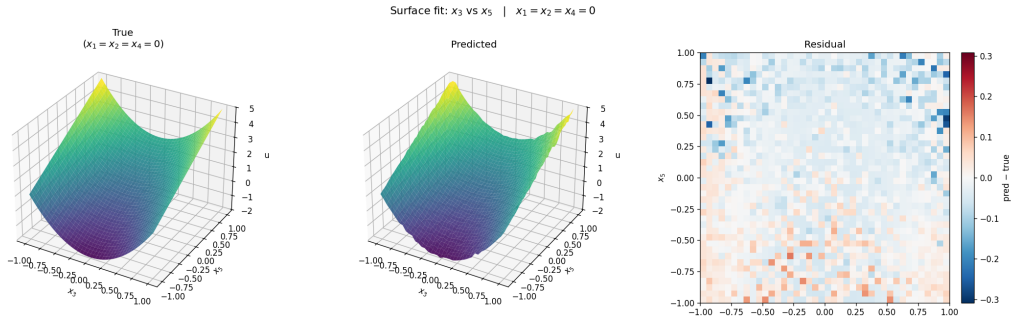


(d) Interaction of x_1 and x_4

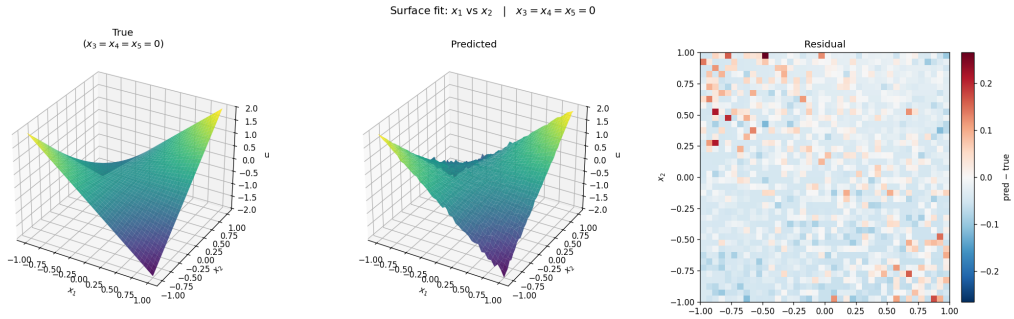
Figure D.8: EFI-Learned marginal and interaction effects for the 5D-Nonlinear example.



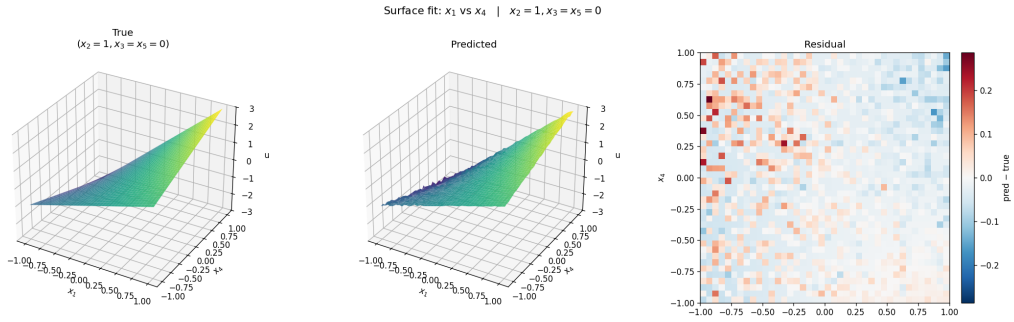
(a) Marginal effects with 95% CIs



(b) Interaction of x_3 and x_5



(c) Interaction of x_1 and x_2



(d) Interaction of x_1 and x_4

Figure D.9: PINN-Learned (λ_{str}=1, dropout 1%) marginal and interaction effects for the 5D-Nonlinear example.

E HYPERPARAMETER SENSITIVITY

This section examines sensitivity to the consistency coefficient, KDE bandwidth, SGLD schedule, temperature, and narrow-neck width. The experiments use Gaussian noise in the Poisson and monotonic regression examples, non-Gaussian noise in the periodic example, and skewed mixture Gaussian errors in the Linear-1D example. Each experiment varies the indicated settings while retaining the remaining configuration described below.

E.1 CONSISTENCY COEFFICIENT

The consistency term prevents the observation-wise inverse outputs from representing n unrelated parameter solutions. Let $k = 1, \ldots, K$ index the retained post-burn-in SGLD/SGD iterations, relabeled for convenience. At iteration k, for the fixed latent state $Z_n^{(k)}$, the zero-consistency condition is

$$\hat{\boldsymbol{\theta}}_1^{(k)} = \cdots = \hat{\boldsymbol{\theta}}_n^{(k)} = \bar{\boldsymbol{\theta}}_n^{(k)},$$

so that one common parameter vector solves the full data-generating system. The dispersion penalized by this term is therefore the within-configuration disagreement across i, not the EFI uncertainty.

EFI uncertainty is obtained across these retained iterations through

$$\left\{ \bar{\boldsymbol{\theta}}_n^{(k)} : k = 1, \ldots, K \right\},$$

and is propagated to $f_{\bar{\boldsymbol{\theta}}_n^{(k)}}(x)$. Thus, $\mathrm{Var}_i(\hat{\boldsymbol{\theta}}_i^{(k)})$ measures failure to reach consensus, whereas the inferential variation is across k in $\bar{\boldsymbol{\theta}}_n^{(k)}$.

η_θ/ϵ can affect the finite-temperature optimization and sampling path. However, in the zero-temperature limit, any $\eta_\theta > 0$ imposes the same consensus condition and therefore leaves the zero-energy feasible set unchanged; unlike the penalty balancing parameter λ_{str} in (3).

Table E.1 reports an ablation for the 1-D Poisson example with Gaussian noise (Appendix D.2). Across the stable schedules—constant $\eta_\theta = 1$ and annealing from 1 to 10 or 100—the MSE remains 0.00030, CI coverage remains 0.9455, CI width remains 0.0685, and PI coverage remains approximately 0.8935. At the same time, the converged consistency residual decreases from 4.9×10^{-5} to 7.0×10^{-8}. Thus, increasing η_θ enforces agreement among the observation-wise inverse outputs more tightly without reducing interval width or calibration across these schedules.

Table E.1: Sensitivity to the consistency scaling parameter η_θ. 1-D Poisson example with Gaussian noise ($\sigma = 0.1$, $n = 100$ observations, $m = 200$ collocation points); mean (SE) over 100 replications at 2×10^5 epochs, with all other settings held fixed. "Annealed" increases η_θ linearly from 1 to the target over the first 20% of training and holds it thereafter. $\mathcal{R}_\theta$ is the consistency term of the energy at convergence.

η_θ schedule	MSE ↓	CI Cov. (95%)	CI Width ↓	PI Cov. (90%)	$\mathcal{R}_\theta$
η_θ=1 (constant)	0.0003 (0.0000)	0.9455 (0.0116)	0.0685 (0.0006)	0.8934 (0.0040)	4.9×10^{-5}
η_θ: 1 → 10 (annealed)	0.0003 (0.0000)	0.9455 (0.0116)	0.0685 (0.0006)	0.8935 (0.0040)	2.6×10^{-6}
η_θ: 1 → 100 (annealed)	0.0003 (0.0000)	0.9455 (0.0116)	0.0685 (0.0006)	0.8936 (0.0040)	7.0×10^{-8}

E.2 KDE BANDWIDTH AND SGLD STEP SIZE

For the KDE bandwidth, Wang & Liang (2026) recommends, for a twice differentiable π_0

$$h_n = c(\log n)^{1/5} h_{\mathrm{ISJ}},$$

where h_{ISJ} is the improved Sheather–Jones plug-in bandwidth (Botev et al., 2010) and $c \geq 1$ controls conservativeness. In the periodic non-Gaussian example, $(\log n)^{1/5} h_{\mathrm{ISJ}} \approx 0.1$, so we consider $h \in \{0.20, 0.25, 0.30\}$. Larger values visibly oversmooth the imputed-error KDE.

With the SGLD schedule fixed at the $h = 0.30$ setting, MSE and PI performance remain stable across bandwidths, while $h = 0.20$ and 0.25 yield CI coverages of 0.883 and 0.915; see Table E.2. Within

the considered range, a larger h generally produces a wider CI because the broader working error distribution is propagated through the learned inverse map G to the extended fiducial distribution of $\boldsymbol{\theta}$. After changing only the initial SGLD step size, the CI coverages increase to 0.954 and 0.948, with essentially unchanged MSE and PI performance; see Table E.3. This indicates that the KDE bandwidth and SGLD sampling dynamics should be considered jointly.

Table E.2: Bandwidth sensitivity with the SGLD schedule fixed at the $h = 0.30$ setting (`poly_1e-4_10.0_0.57`) in the periodic non-Gaussian example.

Bandwidth	MSE ($\times 10^{-3}$)	CI coverage	CI width	PI coverage	PI width
0.20	3.75 (0.33)	0.883 (0.014)	0.225 (0.007)	0.903 (0.003)	1.738 (0.004)
0.25	3.62 (0.28)	0.915 (0.012)	0.235 (0.006)	0.902 (0.003)	1.740 (0.004)
0.30	4.13 (0.27)	0.940 (0.010)	0.297 (0.007)	0.910 (0.003)	1.762 (0.004)

Table E.3: Bandwidth sensitivity after retuning the initial SGLD step size in the periodic non-Gaussian example.

Bandwidth	SGLD schedule	MSE ($\times 10^{-3}$)	CI coverage	CI width	PI coverage	PI width
0.20	`poly_2e-4_10.0_0.57`	3.81 (0.31)	0.954 (0.008)	0.270 (0.007)	0.903 (0.003)	1.739 (0.004)
0.25	`poly_1.75e-4_10.0_0.57`	3.70 (0.28)	0.948 (0.009)	0.259 (0.006)	0.902 (0.003)	1.741 (0.004)

E.3 SGLD Schedule and Temperature

The following experiments use the Linear-1D model with skewed mixture Gaussian errors described in Appendix F.1.

Table E.4 further evaluates SGLD schedules over 100 replications in a separate Linear-1D non-Gaussian example. MSE and PI performance are nearly unchanged across all schedules, whereas schedules with faster decay or smaller initial steps tend to have lower bulk and tail ESS, shorter CIs, and greater undercoverage. ESS is reported only as one convenient quantitative diagnostic of mixing. Its observed association with coverage is illustrative rather than causal and should be interpreted together with trace plots, autocorrelation, energy stabilization, and other diagnostics (Appendix C.4). Overall, these results do not suggest an extensive joint hyperparameter search: we use a small bandwidth grid, adjust the initial SGLD step size only when diagnostics indicate inadequate exploration, retain the default decay parameters, and find the annealing duration insensitive over the tested range.

Table E.4: SGLD schedule ablation in the Linear-1D non-Gaussian example at $h = 0.30$. The step size is $v(\tau) = v_0[1 + (b\tau)^p]^{-1}$, where $\tau = t/T$. Standard errors of the mean are in parentheses.

	SGLD schedule									
Configuration	v_0	b	p	MSE ($\times 10^{-3}$)	$\mathrm{ESS}_{\mathrm{bulk}}$	$\mathrm{ESS}_{\mathrm{tail}}$	CI coverage	CI width	PI coverage	PI width
Reference schedule										
Default	1×10^{-4}	10	0.55	6.10 (0.82)	198 (4)	403 (9)	0.942 (0.018)	0.306 (0.004)	0.904 (0.005)	1.606 (0.011)
Vary initial step v_0 at $b = 10$, $p = 0.55$										
$\frac{1}{2} v_0$	$\mathbf{5 \times 10^{-5}}$	10	0.55	6.17 (0.82)	94 (3)	200 (5)	0.922 (0.021)	0.291 (0.004)	0.904 (0.005)	1.607 (0.011)
$2v_0$	$\mathbf{2 \times 10^{-4}}$	10	0.55	6.08 (0.81)	512 (11)	891 (22)	0.965 (0.014)	0.357 (0.004)	0.903 (0.005)	1.604 (0.011)
Vary decay scale b at $v_0 = 10^{-4}$, $p = 0.55$										
	1×10^{-4}	**20**	0.55	6.12 (0.82)	144 (4)	298 (7)	0.934 (0.020)	0.299 (0.004)	0.904 (0.005)	1.606 (0.011)
	1×10^{-4}	**40**	0.55	6.21 (0.83)	102 (3)	219 (5)	0.924 (0.021)	0.292 (0.004)	0.904 (0.005)	1.605 (0.011)
Vary decay power p at $v_0 = 10^{-4}$, $b = 10$										
	1×10^{-4}	10	**0.75**	6.12 (0.82)	147 (4)	306 (7)	0.935 (0.019)	0.299 (0.004)	0.904 (0.005)	1.606 (0.011)
	1×10^{-4}	10	**0.95**	6.15 (0.82)	108 (3)	232 (6)	0.925 (0.021)	0.294 (0.004)	0.904 (0.005)	1.606 (0.011)

For the temperature ϵ, Liang et al. (2025) found that a sufficiently small fixed value performs reliably in lower-dimensional regression problems. The examples considered here involve more difficult neural-network optimization, so we anneal the inverse temperature $\lambda = 1/\epsilon$ to facilitate early convergence and exploration before reaching a sufficiently low terminal temperature. Table E.5 shows that the terminal temperature should be sufficiently small, whereas the precise annealing duration has

Table E.5: Inverse-temperature ablation in the Linear-1D non-Gaussian example at $h = 0.30$. Here, $\lambda = 1/\epsilon$ is ramped from v_0 to v_1 over a fraction τ of training. Standard errors of the mean over 100 replications are in parentheses.

	λ schedule									
Configuration	v_0	v_1	τ	MSE ($\times 10^{-3}$)	$\mathrm{ESS}_{\mathrm{bulk}}$	$\mathrm{ESS}_{\mathrm{tail}}$	CI coverage	CI width	PI coverage	PI width
Reference schedule										
Default	10	500	0.1	6.10 (0.82)	198 (4)	403 (9)	0.942 (0.018)	0.306 (0.004)	0.904 (0.005)	1.606 (0.011)
Vary terminal inverse temperature v_1 at $\tau = 0.1$										
	10	**200**	0.1	6.10 (0.81)	171 (4)	350 (8)	0.921 (0.021)	0.283 (0.004)	0.905 (0.005)	1.609 (0.011)
	10	**600**	0.1	6.11 (0.82)	213 (5)	427 (9)	0.949 (0.018)	0.317 (0.004)	0.904 (0.005)	1.607 (0.011)
Vary ramp length τ at $v_1 = 500$										
	10	500	**0.2**	6.10 (0.81)	194 (5)	385 (10)	0.944 (0.018)	0.311 (0.004)	0.905 (0.005)	1.609 (0.011)
	10	500	**0.3**	6.12 (0.82)	167 (5)	358 (9)	0.943 (0.018)	0.311 (0.004)	0.906 (0.005)	1.610 (0.011)

little effect over the considered range. Thus, annealing is primarily a computational aid and does not require fine tuning once an adequately low terminal ϵ is reached.

Practical guidance. In practice, we recommend computing $(\log n)^{1/5} h_{\mathrm{ISJ}}$, considering a small conservative grid of c, and excluding bandwidths that visibly oversmooth the imputed-error KDE. For each remaining bandwidth, a small grid of initial SGLD step sizes can be screened using observable mixing and energy diagnostics, together with the additional tools described in Appendix C.4, without using empirical coverage or knowledge of the truth.

E.4 Choice of Narrow-Neck Width

Although the effective dimension q is generally unknown, our ablation results demonstrate empirical robustness to the choice of narrow-neck width d_h over the tested range. Table E.6 reports the corresponding ablation for the 1-D monotonic regression example (Appendix F.3).

Table E.6: Sensitivity to the narrow-neck width d_h for EFI in the 1-D monotonic regression example (encoder $[16, 16, d_h]$). The default configuration uses $d_h = 16$. All other hyperparameters are held fixed at the values in the Monotonic column of Table G.5. Estimation error, CI and PI coverage, and interval widths remain similar across the tested widths $d_h \in \{4, 8, 16\}$, indicating robustness over this range. Results are means over 100 seeds, with standard errors in parentheses. MSE is reported in units of 10^{-4}.

Encoder	Seeds	MSE ($\times 10^{-4}$)	CI Coverage (95%)	CI Range	PI Coverage (90%)	PI Range
$[16, 16, 4]$	100	8.51 (0.35)	0.9313 (0.0053)	0.1046 (0.0005)	0.8835 (0.0042)	0.3280 (0.0018)
$[16, 16, 8]$	100	8.68 (0.33)	0.9356 (0.0045)	0.1051 (0.0006)	0.8840 (0.0040)	0.3279 (0.0019)
$[16, 16, 16]^{\dagger}$	100	8.65 (0.33)	0.9390 (0.0047)	0.1058 (0.0006)	0.8833 (0.0042)	0.3283 (0.0019)

†Paper configuration.

F Additional Numerical Results

This section presents additional numerical experiments. Specifically, we evaluate a 1D linear regression with skewed errors, a second 5-dimensional model adapted from Liang et al. (2018), and a function governed by a non-differentiable monotonicity constraint. These supplementary examples further validate our framework's flexibility, its robust performance in high-dimensional settings, and the fundamental importance of structural constraints as discussed in Section 1.

F.1 1-D Linear Regression with Skewed Mixture Gaussian Errors

We consider a linear regression model governed by the second-order differential equation:

$$\frac{d^2 f}{dx^2} = 0, \quad x \in \Omega = [-1, 1], \tag{F.1}$$

with ground truth $f^\star(x) = 2x - 1$. To examine robustness against non-Gaussianity, we use a skewed mixture noise model $z_i \overset{\text{i.i.d.}}{\sim} 0.8\,\mathcal{N}(-0.25, 0.15^2) + 0.2\,\mathcal{N}(1.0, 0.15^2)$, characterized by well-separated modes and a significant right skew. We use $n = 50$ observations and $m = 200$ PDE collocation points.

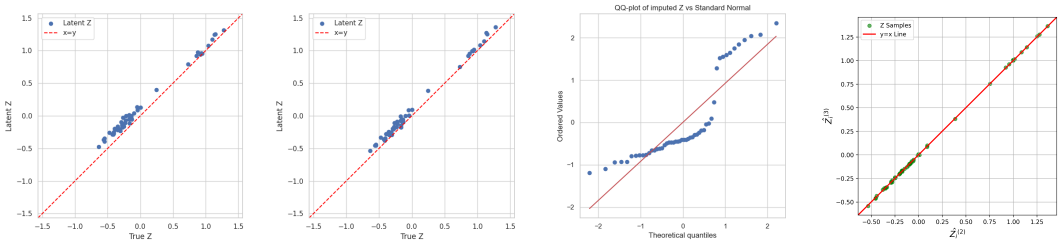
Figure F.2 illustrates the importance of the structural constraint. Without the PDE constraint (F.1) (bottom row), the admissible function class is too large and EFI Run 1 produces highly irregular fits that track local noise patterns. Imposing the constraint reduces the admissible class to linear functions, and the resulting fits are smooth and well-calibrated across both runs.

Convergence diagnostics are shown in Figure F.1. The QQ plot from Run 1 exhibits clear departure from normality, confirming the need for a KDE-based refinement run. After Run 2, the imputed errors align closely with the true latent errors, and the scatter plot of $\boldsymbol{Z}^{(2)}$ vs. $\boldsymbol{Z}^{(3)}$ lies near the diagonal, indicating convergence of the NE-EFI iterations.

Results are reported in Table F.1. All reported MC-dropout PINN configurations under-cover in CI (44.07–71.78%), well below the nominal 95% level. Their Gaussian-working-model PIs also under-cover (85.31–85.68% versus the nominal 90%), with widths of 1.6916–1.7069. EFI attains PI coverage of 90.54% with a narrower interval width of 1.6092. OLS achieves nominal CI coverage but at the cost of over-wide intervals and under-coverage in PI (87.0%), as its Gaussian-based intervals cannot adapt to the asymmetric noise shape. SCP over-covers in PI (93.2%) with substantially wider intervals (2.1704 vs. 1.6092 for EFI): the symmetric nonconformity score used in standard SCP is poorly aligned with the right-skewed noise distribution and compensates by inflating interval widths. EFI alone adapts to the skewed error structure through the KDE refinement: comparing Run 1 and Run 2 in Figure F.2, the intervals shrink appreciably after refinement as the working noise distribution converges to the true skewed shape, yielding the narrowest valid CI and PI while maintaining nominal coverage.

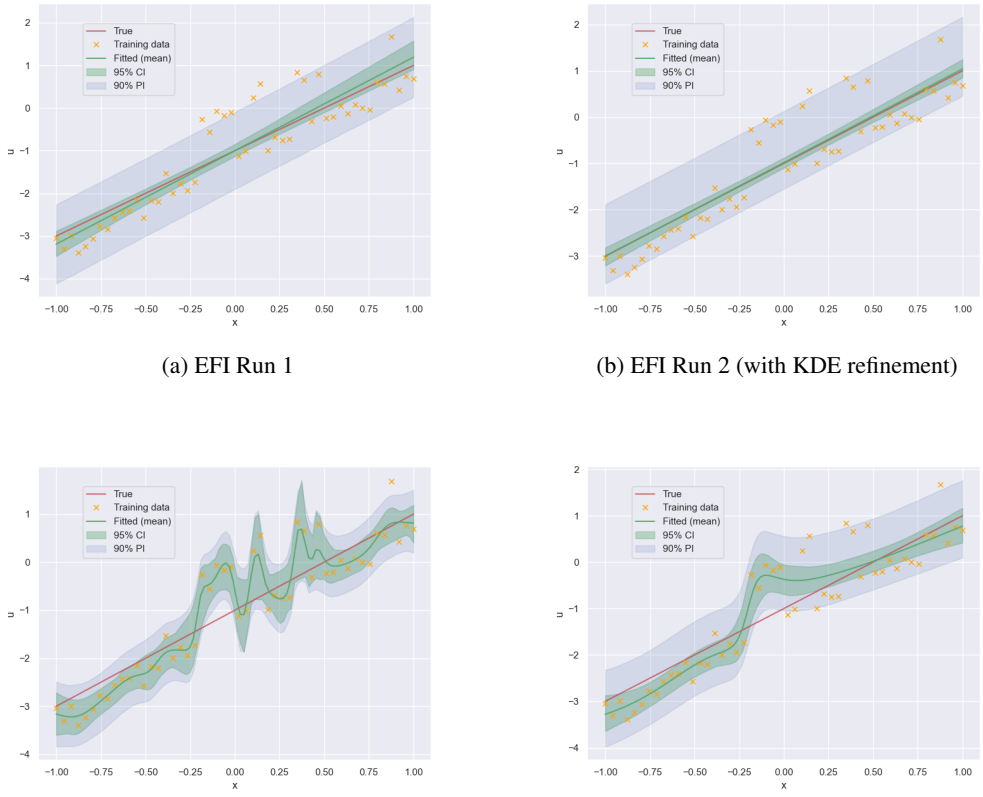
Table F.1: Results on 1-D Linear Example with Skewed Mixture Gaussian Errors

Method	MSE	CI Coverage (95%)	CI Range	PI Coverage (90%)	PI Range
EFI (Ours)	0.0061 (0.0008)	0.9434 (0.0183)	0.3128 (0.0039)	0.9054 (0.0051)	1.6092 (0.0115)
OLS	0.0101 (0.0094)	0.9594 (0.0141)	0.4109 (0.0044)	0.8702 (0.0065)	1.7984 (0.0190)
PINN (λ_{str}=1, dropout 1%)	0.0102 (0.0009)	0.4407 (0.0315)	0.1237 (0.0009)	0.8531 (0.0058)	1.6916 (0.0178)
PINN (λ_{str}=1, dropout 5%)	0.0103 (0.0010)	0.7178 (0.0254)	0.2586 (0.0015)	0.8558 (0.0060)	1.7057 (0.0177)
PINN (λ_{str}=10, dropout 1%)	0.0104 (0.0010)	0.4499 (0.0313)	0.1299 (0.0010)	0.8535 (0.0058)	1.6931 (0.0178)
PINN (λ_{str}=10, dropout 5%)	0.0103 (0.0010)	0.7068 (0.0247)	0.2554 (0.0016)	0.8564 (0.0059)	1.7066 (0.0176)
PINN (λ_{str}=100, dropout 1%)	0.0103 (0.0009)	0.4555 (0.0316)	0.1325 (0.0011)	0.8536 (0.0058)	1.6934 (0.0177)
PINN (λ_{str}=100, dropout 5%)	0.0104 (0.0010)	0.7025 (0.0240)	0.2561 (0.0016)	0.8568 (0.0060)	1.7069 (0.0176)
PINN (λ_{str}=1) + SCP	0.0215 (0.0022)	–	–	0.9318 (0.0060)	2.1704 (0.0354)



(a) EFI Run 1 (b) EFI Run 2 (c) QQ plot (d) $\boldsymbol{Z}^{(2)}$ vs. $\boldsymbol{Z}^{(3)}$

Figure F.1: Latent variable imputation, QQ plot, and convergence diagnostics for the 1-D linear regression case. Note that (a) Run 1: $\boldsymbol{Z}$ vs. $\hat{\sigma}\tilde{\boldsymbol{Z}}^{(1)}$ and (b) Run 2: $\boldsymbol{Z}$ vs. $\boldsymbol{Z}^{(2)}$.



(a) EFI Run 1 (b) EFI Run 2 (with KDE refinement)

(c) EFI Run 1 (d) EFI Run 2 (with KDE refinement)

Figure F.2: 95% CIs and 90% PIs for the 1-D linear example. The top row displays results including PDE constraints; the bottom row displays results without PDE constraints. The comparison contrasts EFI Run 1 with EFI Run 2 (with KDE refinement) in each row.

F.2 LIANG 2018 EXAMPLE

We consider a second high-dimensional example from Liang et al. (2018):

$$f^\star(\boldsymbol{x}) = \frac{5x_1}{1+x_2^2} + 5\sin(x_3x_4) + 2x_5, \quad x_j \in [-1,1], \quad j = 1,\dots,5,$$

with $z_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.1^2)$. We incorporate partial structural information on the linearity of x_1 and x_5:

$$\frac{\partial^2 f}{\partial x_j^2} = 0, \quad j = 1, 5,$$

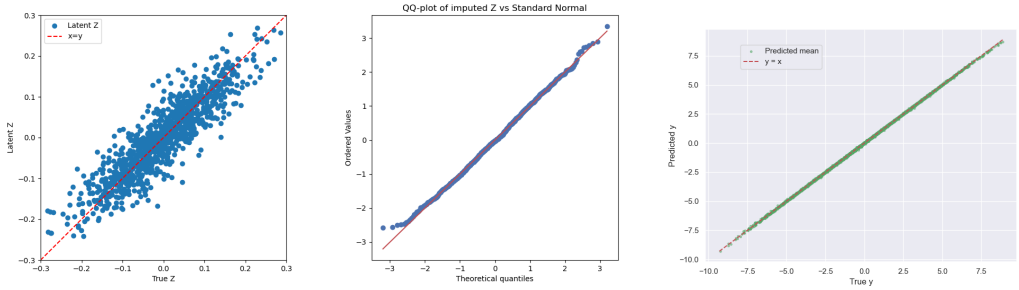
which are verifiable in data exploration phase. We use $n_u = 1{,}000$ observation sensors, $n_f = 500$ PDE collocation points, and $n_{\text{test}} = 5{,}000$ test points drawn uniformly from $[-1,1]^5$. For SS-ANOVA, we include all interactions among x_1, x_2, x_3, x_4 up to order 3, while x_5 is kept linear.

Results are reported in Table F.2. MC-dropout PINNs have MSEs of 0.0108–0.2543, all higher than EFI's MSE of 0.0014. Their CI coverage ranges from 89.04% to 99.22%, with widths of 1.0249–1.8452, indicating sensitivity to the structural-penalty weight and dropout rate. The setting closest to the nominal 95% CI coverage is $\lambda_{\text{str}} = 10$ with dropout 5% (95.76%), but its CI width of 1.7898 is approximately 12.7 times EFI's width of 0.1409. All reported dropout configurations over-cover in PI (96.80–99.26% versus the nominal 90% level). SS-ANOVA severely under-covers (84.4%) even under Gaussian noise. This suggests that the SS-ANOVA uncertainty quantification does not adequately capture the interaction structure, despite competitive point-estimation MSE. EFI attains near-nominal CI coverage (93.55%) with the narrowest intervals among methods that approach the nominal rate. The slight PI under-coverage (87.84%) arises for the same reason as in Section 5.3: the interactions $x_1/(1+x_2^2)$ and $\sin(x_3x_4)$ are not explicitly constrained, leaving excess degrees of freedom in the model.

EFI convergence diagnostics are shown in Figure F.3. The imputed errors align closely with the true errors, the QQ plot confirms the Gaussian working distribution is adequate (no refinement run needed), and the predicted vs. true y scatter lies tightly on the diagonal, confirming accurate point estimation. As in Section 5.3, EFI-learned surfaces are considerably smoother than those of standard PINNs (compare Figures F.4 and F.5): the iterative latent error imputation acts as dynamic data augmentation, preventing overfitting to the static training set and yielding residual plots that are nearly uniformly flat across all interaction surfaces. PINN residual plots, by contrast, exhibit structured patterns indicating systematic misfit, particularly for the x_1–x_2 and x_1–x_5 interactions.

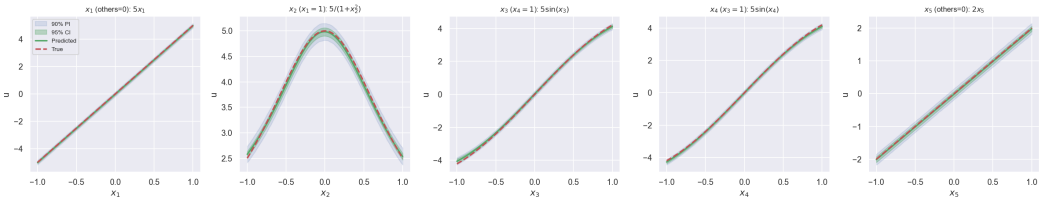
Table F.2: Results on Liang 2018 example with Gaussian noise. All reported MC-dropout PINNs use two hidden layers of 16 units each ([16,16]).

Algorithm	MSE	CI Coverage (95%)	CI Width	PI Coverage (90%)	PI Width
EFI (Ours)	0.0014 (0.0000)	0.9355 (0.0028)	0.1409 (0.0012)	0.8784 (0.0012)	0.3309 (0.0011)
SS-ANOVA	0.0011 (0.0000)	0.8444 (0.0050)	0.0852 (0.0006)	0.8996 (0.0014)	0.3604 (0.0017)
PINN (λ_{str}=1, dropout 1%)	0.0108 (0.0002)	0.9913 (0.0004)	1.0249 (0.0054)	0.9835 (0.0004)	0.8087 (0.0035)
PINN (λ_{str}=1, dropout 5%)	0.0863 (0.0011)	0.9922 (0.0004)	1.8452 (0.0052)	0.9926 (0.0004)	1.7609 (0.0052)
PINN (λ_{str}=10, dropout 1%)	0.0195 (0.0003)	0.9875 (0.0006)	1.1752 (0.0069)	0.9813 (0.0004)	0.9115 (0.0040)
PINN (λ_{str}=10, dropout 5%)	0.1541 (0.0044)	0.9576 (0.0044)	1.7898 (0.0120)	0.9837 (0.0010)	1.9321 (0.0072)
PINN (λ_{str}=100, dropout 1%)	0.0439 (0.0011)	0.9639 (0.0023)	1.3665 (0.0125)	0.9749 (0.0007)	1.1135 (0.0071)
PINN (λ_{str}=100, dropout 5%)	0.2543 (0.0014)	0.8904 (0.0017)	1.7010 (0.0113)	0.9680 (0.0005)	2.1431 (0.0069)
PINN (λ_{str}=10) + SCP	0.0046 (0.0001)	–	–	0.8998 (0.0014)	0.3934 (0.0020)

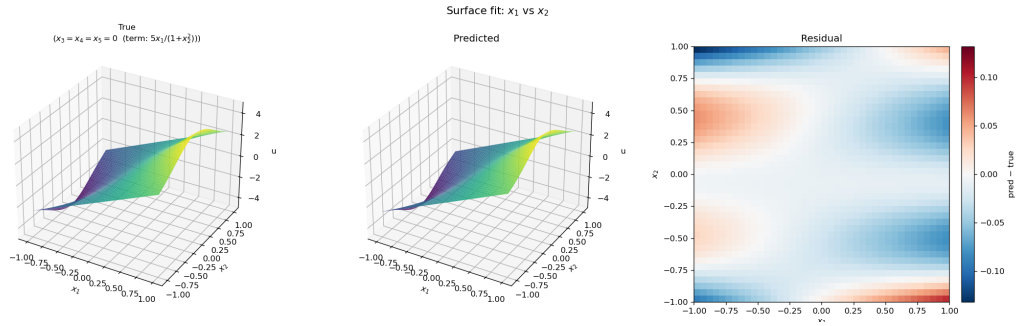


(a) True errors vs imputed errors (b) QQ plot (c) True y vs predicted $\hat{y}$

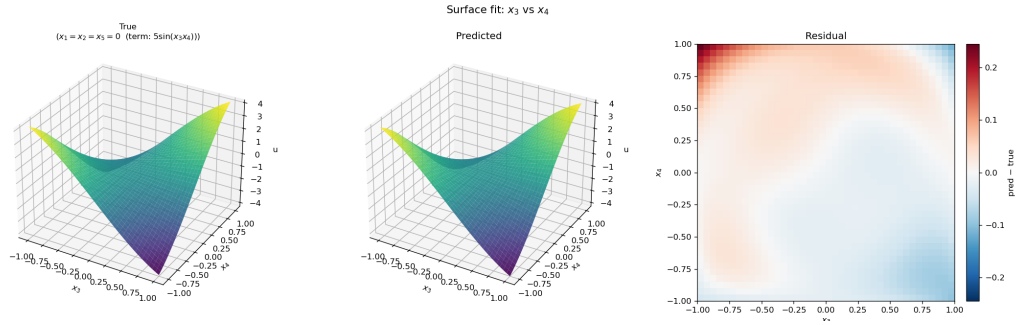
Figure F.3: EFI diagnostic for Liang 2018 example



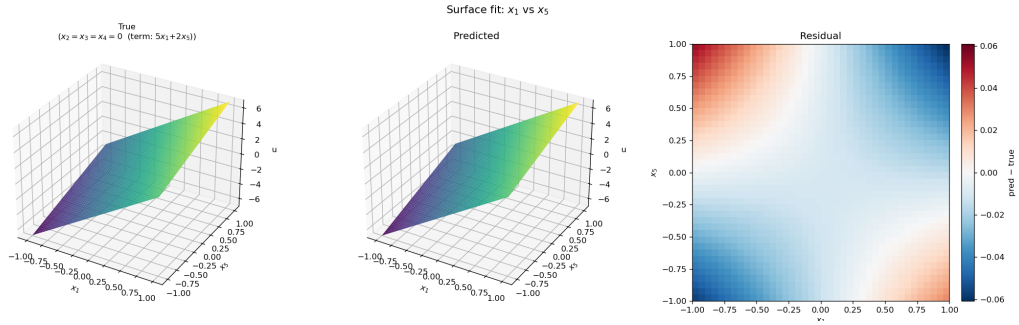
(a) Marginal effects with 95% CIs and 90% PIs



(b) Interaction effects of x_1 and x_2

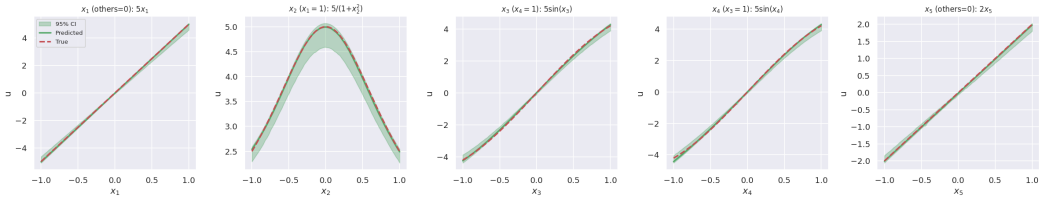


(c) Interaction effects of x_3 and x_4

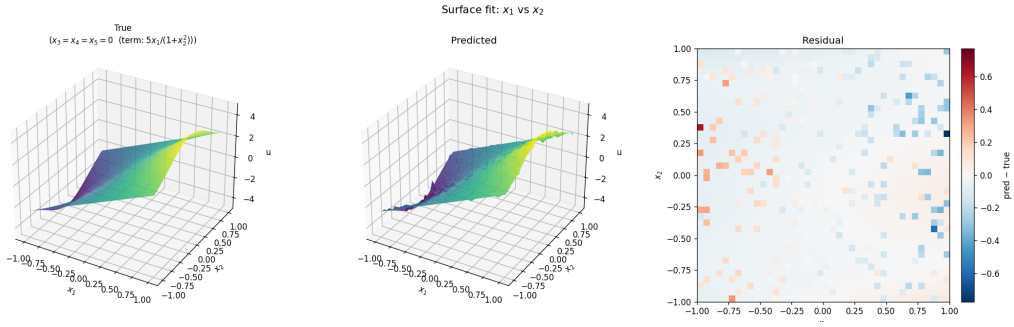


(d) Interaction effects of x_1 and x_5

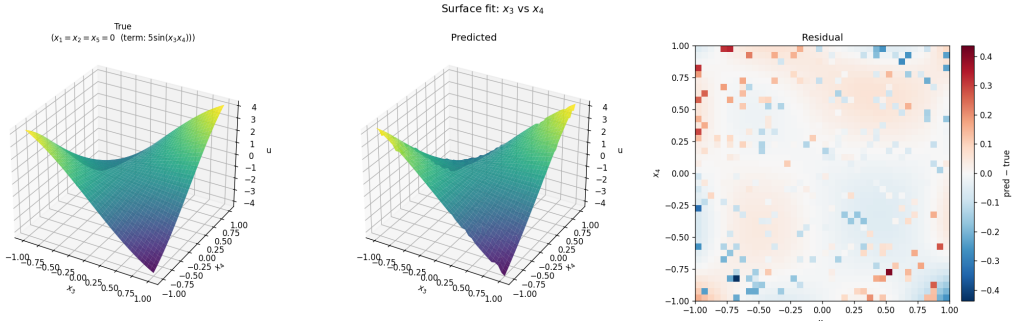
Figure F.4: EFI-Learned marginal and interaction effects for the Liang 2018 example.



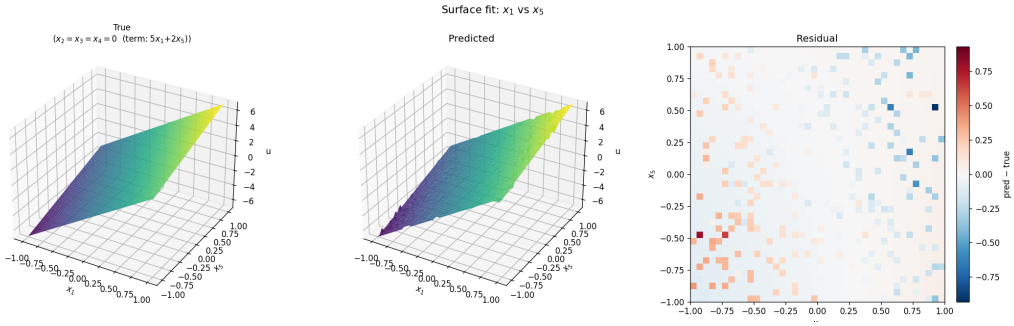
(a) Marginal effects with 95% CIs



(b) Interaction effects of x_1 and x_2



(c) Interaction effects of x_3 and x_4



(d) Interaction effects of x_1 and x_5

Figure F.5: PINN-Learned (λ_{str}=1, dropout 1%) marginal and interaction effects for the Liang 2018 example.

F.3 Monotonic Regression

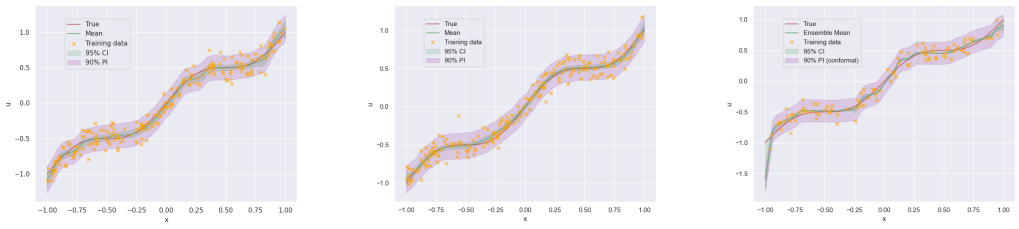
This example provides a second instance of a non-differentiable constraint, complementing the periodic function in Section 5.2. We consider a monotonic regression problem with true solution $f^\star(x) = x + \frac{1}{2\pi}\sin(2\pi x)$ on $x \in [-1, 1]$, satisfying $(f^\star)'(x) = 1 + \cos(2\pi x) \geq 0$ everywhere. Observations follow $y_i = f^\star(x_i) + z_i$ with $n = 200$ and $z_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.1^2)$. To encourage monotonicity, we impose the finite-difference constraint at the specified constraint points; specifically,

$$C^{(1)}(f; \tilde{x}) = f(\tilde{x}) - f(\tilde{x} + \delta) \leq 0, \qquad \delta = 0.05, \tag{F.2}$$

at $m = 200$ collocation points, with violation function $V^{(1)}(c) = \mathrm{ReLU}(c)^2$. This condition requires nonnegative increments over a fixed lag of 0.05 and serves as an approximation to monotonicity; it does not guarantee that the fitted function is nondecreasing throughout the domain.

Table F.3 summarizes the results. MC-dropout PINN intervals remain sensitive to λ_{str} and the dropout rate, with CI coverage ranging from 43.27% to 90.92%, below the nominal 95% level. Their PIs constructed under Gaussian working models also under-cover (82.65–88.72% versus the nominal 90%). Deep ensembles attain the same MSE as EFI at the reported precision (0.0009), but have substantially narrower CIs (0.0372–0.0399 versus 0.1058) and much lower CI coverage (43.55–46.00% versus 93.90%). Their PI coverage is also below nominal (86.27–86.34%).

EFI and BPINN show comparable performance in this example. However, structural constraints in BPINN are still enforced through residual penalties: the constraint-residual scale σ_{diff} controls the balance between data fit and constraint enforcement and must be selected in practice. This scale is distinct from the observation-noise scale σ, which is supplied to BPINN in these experiments but estimated from data by EFI. Figure F.6 illustrates the fitted functions and intervals. Finally, we note that this example is related to shape-constrained regressions (Dümbgen, 2024).



(a) EFI (b) Bayesian PINN (c) Deep ensemble + Conformal

Figure F.6: 95% CI and 90% PI for monotonic example with Gaussian errors.

Table F.3: Results on monotonic model with Gaussian noise.

Algorithm	MSE	CI Coverage (95%)	CI Width	PI Coverage (90%)	PI Width
EFI (Ours)	0.0009 (0.0000)	0.9390 (0.0047)	0.1058 (0.0006)	0.8833 (0.0042)	0.3283 (0.0019)
PINN (λ_{str}=1, dropout 1%)	0.0015 (0.0001)	0.6974 (0.0077)	0.0737 (0.0005)	0.8265 (0.0046)	0.2940 (0.0015)
PINN (λ_{str}=1, dropout 5%)	0.0011 (0.0000)	0.9092 (0.0056)	0.1310 (0.0004)	0.8715 (0.0043)	0.3240 (0.0014)
PINN (λ_{str}=10, dropout 1%)	0.0011 (0.0001)	0.6365 (0.0099)	0.0598 (0.0003)	0.8489 (0.0042)	0.3040 (0.0015)
PINN (λ_{str}=10, dropout 5%)	0.0011 (0.0001)	0.8096 (0.0085)	0.1235 (0.0004)	0.8844 (0.0040)	0.3356 (0.0015)
PINN (λ_{str}=100, dropout 1%)	0.0013 (0.0000)	0.5228 (0.0067)	0.0538 (0.0002)	0.8713 (0.0039)	0.3264 (0.0016)
PINN (λ_{str}=100, dropout 5%)	0.0071 (0.0001)	0.4327 (0.0052)	0.1142 (0.0004)	0.8872 (0.0038)	0.4205 (0.0015)
PINN (λ_{str}=10) + SCP	0.0050 (0.0007)	–	–	0.9017 (0.0039)	0.3862 (0.0035)
Deep Ensemble (λ_{str}=100, M=20)	0.0009 (0.0000)	0.4382 (0.0100)	0.0383 (0.0006)	0.8630 (0.0043)	0.3109 (0.0016)
Deep Ensemble (λ_{str}=1000, M=20)	0.0009 (0.0000)	0.4355 (0.0099)	0.0372 (0.0006)	0.8627 (0.0042)	0.3111 (0.0016)
Deep Ensemble (λ_{str}=1000, M=50)	0.0009 (0.0000)	0.4600 (0.0103)	0.0399 (0.0006)	0.8634 (0.0041)	0.3116 (0.0015)
BPINN (σ_{diff}=0.1)	0.0007 (0.0000)	0.9287 (0.0058)	0.0910 (0.0007)	0.8966 (0.0032)	0.3376 (0.0002)
BPINN (σ_{diff}=0.01)	0.0006 (0.0000)	0.9292 (0.0061)	0.0842 (0.0005)	0.8963 (0.0033)	0.3363 (0.0001)
BPINN (σ_{diff}=0.001)	0.0006 (0.0000)	0.9124 (0.0068)	0.0818 (0.0004)	0.8960 (0.0032)	0.3359 (0.0001)

G EXPERIMENTAL SETTINGS

In this section, we provide detailed experiment settings to ensure better reproducibility. We note that $\lambda = 1/\epsilon$ and η_{str} is λ_{str} in the following tables for EFI.

G.1 MC DROPOUT AND DEEP ENSEMBLE BASELINES

Algorithm G.1: MC Dropout PINN (Gal & Ghahramani, 2016)

Input: Data D, structural constraints, dropout rate p, structural weight λ_{str}, weight decay γ, training iterations E, MC size T, and evaluation points $\{x_m^\star\}_{m=1}^M$.

Initialize θ_0.

for $t = 1, \ldots, E$ **do**

Draw dropout mask ω_t and update θ_t by Adam using $\mathcal{L}_t$ defined as

$$\mathcal{L}_t(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f_\theta^{\omega_t}(x_i)\right)^2 + \lambda_{\text{str}}\sum_{\ell=1}^{L}\frac{1}{m_\ell}\sum_{j=1}^{m_\ell} V^{(\ell)}\left(C^{(\ell)}(f_\theta^{\omega_t}; \tilde{x}_j^{(\ell)})\right) + \gamma\|\theta\|_2^2, \qquad \text{(G.1)}$$

where $f_\theta^{\omega_t}(x_i)$ denotes a DNN with dropout mask ω_t on.

Set $\hat{\theta} \leftarrow \theta_E$ and freeze the network weights.

for $t = 1, \ldots, T$ **do**

Draw $\omega^{(t)}$ independently and compute $f^{(t)}(x) = f_{\hat{\theta}}^{\omega^{(t)}}(x)$ on the training and evaluation points.

Compute $\hat{f}_{\text{MC}}(x) = T^{-1}\sum_{t=1}^{T} f^{(t)}(x)$ and construct $\text{CI}(x^\star)$ from the empirical quantiles of $\{f^{(t)}(x^\star)\}_{t=1}^T$.

Estimate $\hat{\sigma}^2 = n^{-1}\sum_{i=1}^{n}\{y_i - \hat{f}_{\text{MC}}(x_i)\}^2$.

for $t = 1, \ldots, T$ **do**

Draw $\epsilon^{(t)} \sim N(0,1)$ and set $Y^{\star(t)}(x^\star) = f^{(t)}(x^\star) + \hat{\sigma}\epsilon^{(t)}$.

Construct $\text{PI}(x^\star)$ from the empirical quantiles of $\{Y^{\star(t)}(x^\star)\}_{t=1}^T$.

return $\hat{f}_{\text{MC}}$, $\hat{\sigma}$, CI, and PI.

Algorithm G.2: Deep Ensemble PINN with Optional Subagging (Lakshminarayanan et al., 2017)

Input: Data $D = \{(x_i, y_i)\}_{i=1}^{n}$, structural penalty R_{str}, weight λ_{str}, epochs E, ensemble size M, subsample ratio $s \in (0, 1]$, predictive draws K, and miscoverage levels $\alpha_{\text{CI}}, \alpha_{\text{PI}}$.

for $r = 1, \ldots, M$ **do**

Draw $D_r \subseteq D$ uniformly without replacement with $n_r = \max\{1, \text{round}(sn)\}$; set $D_r = D$ if $s = 1$.

Independently initialize θ_r and train for E epochs using Adam to minimize

$$\mathcal{L}_r(\theta) = \sum_{i \in D_r} \{y_i - f_\theta(x_i)\}^2 + \lambda_{\text{str}} R_{\text{str}}(f_\theta),$$

yielding $\hat{\theta}_r$.

Set

$$\hat{\sigma}_r^2 = \frac{1}{n_r} \sum_{i \in D_r} \{y_i - f_{\hat{\theta}_r}(x_i)\}^2.$$

for *each* $x^\star$ **do**

Compute

$$\hat{f}_{\text{ens}}(x^\star) = \frac{1}{M} \sum_{r=1}^{M} f_{\hat{\theta}_r}(x^\star),$$

and

$$\text{CI}(x^\star) = \left[\hat{q}_{\alpha_{\text{CI}}/2}, \hat{q}_{1-\alpha_{\text{CI}}/2}\right] \{f_{\hat{\theta}_r}(x^\star)\}_{r=1}^{M}.$$

Draw $Z_{rk} \overset{\text{iid}}{\sim} N(0, 1)$ and form

$$Y_{rk}^\star = f_{\hat{\theta}_r}(x^\star) + \hat{\sigma}_r Z_{rk}, \qquad r = 1, \ldots, M, \quad k = 1, \ldots, K.$$

Construct

$$\text{PI}(x^\star) = \left[\hat{q}_{\alpha_{\text{PI}}/2}, \hat{q}_{1-\alpha_{\text{PI}}/2}\right] \{Y_{rk}^\star\}_{r,k}.$$

return $\hat{f}_{\text{ens}}$, CI, PI.

G.2 Learning-Rate Schedules

All schedules are parameterized by a normalized training progress $t \in [0, 1]$, where $t = \text{epoch}/\text{total_epochs}$.

Table G.1: Schedule types used for hyperparameter annealing.

Name	**Formula** $f(t)$	**Description**
Constant	v	Fixed value throughout training.
Linear	$v_0 + (v_1 - v_0)\, t$	Linear interpolation from v_0 at t=0 to v_1 at t=1.
Log (geometric)	$v_0 \left(\dfrac{v_1}{v_0}\right)^t$	Exponential (geometric) interpolation in log-space.
Polynomial	$\dfrac{v_0}{1 + (\lambda\, t)^p}$	Inverse-polynomial decay; $\lambda > 0$ is a scale and $p > 0$ is the exponent.

Schedules may be *composed* by chaining segments with the + operator and assigning each segment a progress breakpoint τ_i via `@`. Segment i runs over the sub-interval $[\tau_{i-1}, \tau_i)$ with its own local progress rescaled to $[0, 1]$. The final segment's terminal value is held constant for any remaining progress. For example, the string `lin_`v_0`_`v_1`@0.5+const_`v_2 applies a linear warm-up over the first half of training, then holds v_2 for the second half.

Lastly, we note that some non-Gaussian experiments adopt different SGD/SGLD learning rates in step 2 and 3, compared to step 1. This is due to the nature of sampling: in step 1, EFI samples from a Gaussian; however, in step 2 and beyond, EFI samples from an fitted KDE. Such distribution difference sometimes requires different step sizes when sampling/optimizing.

Table G.2: Hyperparameter configurations for EFI on the Linear-1D and Poisson models.

Hyperparameter	**Linear-1D**	**Poisson**
Data Generation		
Domain Ω	$[-1.0,\ 1.0]$	$[-0.7,\ 0.7]$
Observation sensors n_u	50	100
PDE sensors n_f	200	200
sol_sd σ_u	1.0	1.0
Noise distribution	`mixture_of_gaussians`	`mixture_of_gaussians`
Component means	$[-0.25,\ 1.0]$	$[-0.6,\ 0.6]$
Component stds	$[0.15,\ 0.15]$	$[0.2,\ 0.2]$
Component weights	$[0.8,\ 0.2]$	$[0.5,\ 0.5]$
Training		
Epochs	200,000	200,000
Burn-in ratio	0.2	0.2
Save every	1	1
PINN Architecture		
Network architecture	`dnn`	`dnn`
Hidden layers	[30, 30]	[50, 50]
Activation function	`tanh`	`tanh`
k	–	–
Optimization		
Learning rate (SGD)	`poly_3e-6_10.0_0.55`	`poly_5e-7_10.0_0.75`
SGD momentum	0.9	0.9
Learning rate (SGLD)	`poly_1e-4_10.0_0.55`	`poly_2e-4_10.0_0.95`
SGHMC α	0.1	0.1
Loss Weights		
λ	`log_10.0_500.0@0.1`	`linear_10.0_500.0@0.1`
λ_{PDE}	`linear_1.0_5.0@0.1`	`linear_1.0_10.0@0.1/linear_1.0_5.0@0.1/1.0`
λ_θ	1.0	1.0
Encoder		
Hidden layers	[16, 8, 4]	[16, 8, 4]
Activation function	`leaky_relu`	`leaky_relu`
NE-EFI		
Step	1/2/3	1/2/3
Bandwidth h	None/0.3/0.3	None/0.3/0.3

Table G.3: Hyperparameter configuration for EFI on the Nonlinear-5D and Liang2018 models.

Hyperparameter	**Nonlinear-5D**	**Liang2018**
Data Generation		
Domain Ω	$[-1.0, 1.0]^5$	$[-1.0, 1.0]^5$
Observation sensors n_u	500	1,000
PDE sensors n_f	200	500
sol_sd σ_u	0.1	0.1
Noise distribution	`gaussian`	`gaussian`
Training		
Epochs	200,000	200,000
Burn-in ratio	0.2	0.2
Save every	50	50
PINN Architecture		
Network architecture	`dnn`	`dnn`
Hidden layers	[50, 50, 50]	[16, 16]
Activation function	`tanh`	`tanh`
k	–	–
Optimization		
Learning rate (SGD)	`poly_5e-7_10.0_0.75`	`poly_9e-7_80.0_0.75`
SGD momentum	0.9	0.9
Learning rate (SGLD)	`poly_1e-4_10.0_0.95`	`poly_1.5e-4_50.0_0.95`
SGHMC α	0.1	0.1
Loss Weights		
λ	`linear_10.0_1100.0`	`linear_10.0_800.0`
λ_{PDE}	1.0	`linear_1.0_30.0@0.1`
λ_θ	1.0	1.0
Encoder		
Hidden layers	[32, 32, 16]	[16, 16, 8]
Activation function	`leaky_relu`	`leaky_relu`
NE-EFI		
Step	1	1
Bandwidth h	None	None

Table G.4: Hyperparameter configurations for EFI on the Periodic models with Gaussian and Non-Gaussian noise.

Hyperparameter	**Periodic (Gaussian)**	**Periodic (Non-Gaussian)**
Data Generation		
Domain Ω	$[-0.5, 0.5]$	$[-0.5, 0.5]$
Observation sensors n_u	300	300
PDE sensors n_f	200	200
sol_sd σ_u	0.1	1.0
Noise distribution	`gaussian`	`mixture_of_gaussians`
Component means	–	$[-0.6, 0.6]$
Component stds	–	$[0.2, 0.2]$
Component weights	–	$[0.5, 0.5]$
Period	–	0.5
Training		
Epochs	300,000	300,000
Burn-in ratio	0.2	0.2
Save every	50	50
PINN Architecture		
Network architecture	`dnn`	`dnn`
Hidden layers	[30, 30, 30]	[30, 30, 30]
Activation function	`relu`	`relu`
k	–	–
Optimization		
Learning rate (SGD)	`poly_1.1e-6_50.0_0.57`	`poly_1.1e-6_50.0_0.57`
Learning rate (SGD, Steps 2 & 3)	–	`poly_1e-6_10.0_0.57`
SGD momentum	`const_0.9@0.9+const_0.0`	`const_0.9@0.9+const_0.0`
Learning rate (SGLD)	`poly_1e-4_10.0_0.57`	`poly_1e-4_10.0_0.57`
SGHMC α	0.1	0.1
Loss Weights		
λ	`linear_10.0_1200.0`	`linear_10.0_1200.0`
λ_{PDE}	`const_0.0@0.1+linear_0.0_1.0@0.2`	`const_0.0@0.1+linear_0.0_1.0@0.2`
λ_θ	1.0	1.0
Encoder		
Hidden layers	[16, 8, 4]	[16, 8, 4]
Activation function	`leaky_relu`	`leaky_relu`
NE-EFI		
Step	1	1/2/3
Bandwidth h	None	None/0.3/0.3

Table G.5: Hyperparameter configurations for EFI on the Poisson (Gaussian) and Monotonic models.

Hyperparameter	**Poisson (Gaussian)**	**Monotonic**
Data Generation		
Domain Ω	$[-0.7,\ 0.7]$	$[-1.0,\ 1.0]$
Observation sensors n_u	100	200
PDE sensors n_f	200	200
sol_sd σ_u	0.1	0.1
Noise distribution	`gaussian`	`gaussian`
Constraint mode	–	`value`
Boundary shift	–	0.05
Training		
Epochs	200,000	300,000
Burn-in ratio	0.2	0.2
Save every	1	50
PINN Architecture		
Network architecture	`dnn`	`dnn`
Hidden layers	[50, 50]	[30, 30, 30]
Activation function	`tanh`	`relu`
Optimization		
Learning rate (SGD)	`poly_7e-7_10.0_0.75`	`poly_1e-6_50.0_0.57@0.2`
SGD momentum	0.9	0.9
Learning rate (SGLD)	`poly_2e-4_10.0_0.75`	`poly_1e-4_20.0_0.57@0.2`
SGHMC α	0.1	0.1
Loss Weights		
λ	`linear_10.0_800.0@0.1`	`linear_100.0_1500.0@0.2`
λ_{PDE}	`0.25/0.5/1.0`	`const_0.0@0.1+linear_0.0_1.0@0.2`
λ_θ	1.0	1.0
Encoder		
Hidden layers	[16, 8, 4]	[16, 16, 16]
Activation function	`leaky_relu`	`leaky_relu`
NE-EFI		
Step	1	1
Bandwidth	None	None

Table G.6: Deep Ensemble hyperparameters for the 1-D Poisson example with bimodal Gaussian-mixture errors.

Hyperparameter	**Poisson (non-Gaussian)**
Data Generation	
Domain Ω	$[-0.7,\ 0.7]$
Observation sensors n_u	100
PDE sensors n_f	200
`sol_sd` σ_u	1.0
Noise distribution	`mixture_of_gaussians`
Component means	$[-0.6,\ 0.6]$
Component stds	$[0.2,\ 0.2]$
Component weights	$[0.5,\ 0.5]$
Effective noise sd	0.632
Training (per member)	
Epochs	20,000
Burn-in ratio	0.2
Save every	50
Member Architecture	
Network architecture	`dnn`
Hidden layers	[50, 50]
Activation function	`tanh`
Optimization	
Learning rate (Adam)	3×10^{-4}
Weight decay λ_w	0.0
Loss Weights	
λ_{str}	10 / 100 / 1000
Ensemble	
Ensemble size M	50
Subsample ratio s	1.0 (DE) / 0.8 (DE+Sub)
Gaussian draws per member K	100

Table G.7: Deep Ensemble hyperparameters for the Gaussian-error examples.

Hyperparameter	**Poisson**	**Periodic**	**Monotonic**	**Nonlinear-5d**
Data Generation				
Domain Ω	$[-0.7,\ 0.7]$	$[-0.5,\ 0.5]$	$[-1.0,\ 1.0]$	$[-1.0,\ 1.0]^5$
Input dimension d	1	1	1	5
Observation sensors n_u	100	300	200	500
PDE sensors n_f	200	200	200	200
`sol_sd` σ_u	0.1	0.1	0.1	0.1
`diff_sd`	—	—	0.0	0.0
Replicates (n_u / n_f)	—	—	1 / 1	1 / 1
Noise distribution	`gaussian`	`gaussian`	`gaussian`	`gaussian`
Period	—	0.5	—	—
Constraint mode	—	—	`value` (δ=0.05)	—
Training (per member)				
Epochs	20,000	20,000	20,000	20,000
Burn-in ratio	0.2	0.2	0.2	0.2
Save every	50	50	50	50
Member Architecture				
Network architecture	`dnn`	`dnn`	`dnn`	`dnn`
Hidden layers	[50, 50]	[30, 30, 30]	[30, 30, 30]	[50, 50, 50]
Activation function	`tanh`	`relu`	`relu`	`tanh`
Optimization				
Learning rate (Adam)	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Weight decay λ_w	0.0	0.0	0.0	0.0
Loss Weights				
λ_{str}	10 / 100 / 1000	1 / 10 / 100 / 1000	100 / 1000	10 / 100 / 1000
Ensemble				
Ensemble size M	50	50	20 / 50	50
Subsample ratio s	1.0	1.0	1.0	1.0
Gaussian draws per member K	100	100	100	100

Table G.8: Hyperparameter configurations for Dropout PINN on the Linear-1D, Poisson, and Periodic models with non-Gaussian noise.

Hyperparameter	**Linear-1D**	**Poisson**	**Periodic**
Data Generation			
Domain Ω	$[-1.0,\ 1.0]$	$[-0.7,\ 0.7]$	$[-0.5,\ 0.5]$
Observation sensors n_u	50	100	300
PDE sensors n_f		200	
`sol_sd` σ_u		1.0	
Noise distribution		`mixture_of_gaussians`	
Component means	$[-0.25,\ 1.0]$	$[-0.6,\ 0.6]$	$[-0.6,\ 0.6]$
Component stds	$[0.15,\ 0.15]$	$[0.2,\ 0.2]$	$[0.2,\ 0.2]$
Component weights	$[0.8,\ 0.2]$	$[0.5,\ 0.5]$	$[0.5,\ 0.5]$
Effective noise sd	0.522	0.632	0.632
Model-specific	–	–	period $= 0.5$
Training			
Epochs	200,000	200,000	300,000
Burn-in ratio		0.2	
Save every		50	
PINN Architecture			
Network architecture		`dropout`	
Hidden layers	[30, 30]	[50, 50]	[30, 30, 30]
Activation function	`tanh`	`tanh`	`relu`
Dropout rate p		0.01 / 0.05	
k		–	
Optimization			
Learning rate (Adam)		3×10^{-4}	
Weight decay λ_w		1×10^{-5}	
Loss Weights			
λ_{PDE}		1.0 / 10.0 / 100.0	
Uncertainty Quantification			
MC dropout samples T		1,000	

Table G.9: Hyperparameter configurations for Dropout PINN on Liang2018, Nonlinear-5D, Periodic, and Monotonic models with Gaussian noise.

Hyperparameter	**Liang2018**	**Nonlinear-5D**	**Periodic**	**Monotonic**
Data Generation				
Domain Ω	$[-1.0,\ 1.0]^5$	$[-1.0,\ 1.0]^5$	$[-0.5,\ 0.5]$	$[-1.0,\ 1.0]$
Observation sensors n_u	1000	500	300	200
PDE sensors n_f	500	200	200	200
`sol_sd` σ_u	0.1			
Noise distribution	`gaussian`			
Component means	–			
Component stds	–			
Component weights	–			
Model-specific	–	–	–	`value`, shift $= 0.05$
Training				
Epochs	200,000	200,000	300,000	200,000
Burn-in ratio	0.2			
Save every	50			
PINN Architecture				
Network architecture	`dropout`			
Hidden layers	[16, 16]	[50, 50, 50]	[30, 30, 30]	[30, 30, 30]
Activation function	`tanh`	`tanh`	`relu`	`relu`
Dropout rate p	0.01 / 0.05			
k	–			
Optimization				
Learning rate (Adam)	3×10^{-4}			
Weight decay λ_w	1×10^{-5}			
Loss Weights				
λ_{PDE}	1.0 / 10.0 / 100.0			
Uncertainty Quantification				
MC dropout samples T	1,000			

Table G.10: Hyperparameter configurations for SCP on Linear-1D, Poisson, and Periodic (Non-Gaussian) models.

Hyperparameter	**Linear-1D**	**Poisson**	**Periodic (Non-Gaussian)**
Data Generation			
Domain Ω	$[-1.0,\ 1.0]$	$[-0.7,\ 0.7]$	$[-0.5,\ 0.5]$
Observation sensors n_u	50	100	300
PDE sensors n_f	200	200	200
sol_sd σ_u	1.0	1.0	1.0
Noise distribution	`mixture_of_gaussians`		
Component means	$[-0.25,\ 1.0]$	$[-0.6,\ 0.6]$	$[-0.6,\ 0.6]$
Component stds	$[0.15,\ 0.15]$	$[0.2,\ 0.2]$	$[0.2,\ 0.2]$
Component weights	$[0.8,\ 0.2]$	$[0.5,\ 0.5]$	$[0.5,\ 0.5]$
Training			
Epochs	200,000	200,000	300,000
Burn-in ratio	0.2		
Save every	1	1	50
PINN Architecture			
Network architecture	`dnn`		
Hidden layers	[30, 30]	[50, 50]	[30, 30, 30]
Activation function	`tanh`	`tanh`	`relu`
Optimization			
Learning rate (Adam)	3×10^{-4}		
Loss Weights			
λ_{PDE}	1.0	100.0	10.0
Conformal Calibration			
Calibration ratio	0.5		
Miscoverage level α	0.1		

Table G.11: Hyperparameter configurations for SCP on Liang2018, Nonlinear-5D, and Periodic (Gaussian) models.

Hyperparameter	**Liang2018**	**Nonlinear-5D**	**Periodic (Gaussian)**
Data Generation			
Domain Ω	$[-1.0,\ 1.0]^5$	$[-1.0,\ 1.0]^5$	$[-0.5,\ 0.5]$
Observation sensors n_u	1000	500	300
PDE sensors n_f	500	200	200
sol_sd σ_u		0.1	
Noise distribution		`gaussian`	
Training			
Epochs	200,000	200,000	300,000
Burn-in ratio		0.2	
Save every		50	
PINN Architecture			
Network architecture		`dnn`	
Hidden layers	[16, 16]	[50, 50, 50]	[30, 30, 30]
Activation function	`tanh`	`tanh`	`relu`
Optimization			
Learning rate (Adam)		3×10^{-4}	
Loss Weights			
λ_{PDE}	10.0	10.0	1.0
Conformal Calibration			
Calibration ratio		0.5	
Miscoverage level α		0.1	

Table G.12: Hyperparameter configurations for Dropout PINN and Split Conformal Prediction (SCP) on the Monotonic model.

Hyperparameter	**Monotonic (Dropout PINN)**	**Monotonic (SCP)**
Data Generation		
Domain Ω	$[-1.0,\ 1.0]$	
Observation sensors n_u	200	
PDE sensors n_f	200	
sol_sd σ_u	0.1	
Noise distribution	`gaussian`	
Constraint mode	`value`	
Shift	0.05	
Training		
Epochs	200,000	
Burn-in ratio	0.2	
Save every	50	
PINN Architecture		
Network architecture	`dnn`	
Hidden layers	[30, 30, 30]	
Activation function	`relu`	
Dropout rate	0.01/0.05	–
Optimization		
Learning rate (Adam)	3×10^{-4}	
Loss Weights		
λ_{PDE}	1.0/10.0/100.0	10.0
Conformal Calibration		
Calibration ratio	–	0.5
Miscoverage level α	–	0.1