

# Bayesian Physics-Informed Extreme Learning Machine for Forward and Inverse PDE Problems with Noisy Data

Xu Liu<sup>a</sup>, Wen Yao<sup>a,\*</sup>, Wei Peng<sup>a,\*</sup>, Weien Zhou<sup>a</sup>

<sup>a</sup>*Defense Innovation Institute, Chinese Academy of Military Science, Beijing 100071, China*

---

## Abstract

Physics-informed extreme learning machine (PIELM) has recently received significant attention as a rapid version of physics-informed neural network (PINN) for solving partial differential equations (PDEs). The key characteristic is to fix the input layer weights with random values and use Moore–Penrose generalized inverse for the output layer weights. The framework is effective, but it easily suffers from overfitting noisy data and lacks uncertainty quantification for the solution under noise scenarios. To this end, we develop Bayesian physics-informed extreme learning machine (BPIELM) to solve both forward and inverse linear PDE problems with noisy data in a unified framework. In our framework, a prior probability distribution is introduced in the output layer for extreme learning machine with physic laws and the Bayesian method is used to estimate the posterior of parameters. Besides, for inverse PDE problems, problem parameters considered as new output layer weights are unified in a framework with forward PDE problems. Finally, we demonstrate BPIELM considering both forward problems, including Poisson, advection, and diffusion equations, as well as inverse problems, where unknown problem parameters are estimated. The results show that, compared with PIELM, BPIELM quantifies uncertainty arising from noisy data and provides more accurate predictions. In addition, BPIELM is considerably cheaper than PINN in terms of the computational cost.

*Keywords:* Partial differential equation, Uncertainty quantification, Extreme learning machine, Physics-informed neural network

---

\*Corresponding author

*Email addresses:* liuxu18054448691@126.com (Xu Liu), Wendy0782@126.com (Wen Yao), weipeng0098@126.com (Wei Peng)

## 1. Introduction

How to leverage the existing data to accurately model and predict PDE-based systems remains an important issue, especially for solving ill-posed problems, e.g., with unknown parameters and boundary conditions, for alleviating the influence by noisy data collected from different sensors, and for reducing the computational cost [1, 2]. Recently, physics-informed machine learning, as a powerful tool, has attracted increasing attention to addressing those challenges [3]. In particular, physics-informed neural network (PINN), a general framework for solving both forward and inverse PDE problems, encodes the underlying physics laws into loss function to constrain the space of admissible solutions and makes predictions for unknown terms given limited data [4, 5, 6]. PINN has been successfully used for solving PDEs or complex PDE-based physics problems in various domains, such as materialogy [7], medical diagnosis [8] and hidden fluid mechanics [9], etc.

Quantifying prediction uncertainty associated with, such as noisy data and unknown terms, is important for neural network (NN) based inference [10, 11]. However, PINN without the built-in uncertainty quantification (UQ) may restrict physical and biological applications that require high reliability. For some UQ problems, generating distributions of output predictions is more important than point estimates of the solution [12]. In this regard, Yang et al. [13, 14] quantify and propagate uncertainty in systems by combining generative adversarial network (GAN) and physics-based loss function. The latent variable models are constructed as probabilistic representations and the adversarial inference procedure is utilized to train models. While physics laws of systems are encoded into the loss function instead of the discriminator. To exploit the full potential of the adversarial inference procedure, Daw et al. [15] propose a GAN framework based on a physics-informed discriminator, where the physics laws are incorporated into the learning of both the generator and the discriminator model. In addition, the Bayesian method achieves many successful research efforts about UQ in deep learning [16, 17, 18]. Yang et al. [19] combine the Bayesian neural network and PINN to provide predictions and quantify the uncertainty given noisy data. Prior probability distributions are placed over weights and the inference is then performed on weights, while the inference is not tractable in general. Therefore, variational inference is often used to approximate the inference and these models require more parameters and more computational cost to converge. To this end, dropout is utilized to estimate the uncertainty in [20], where arbitrary polynomial chaos is combined with PINN to solve stochastic PDE problems. In current researches, GAN, Bayesian method or dropout is integrated with PINN to quantify prediction uncertainty,

where PINN is the basic network structure. PINN is particularly promising for ill-posed forward and inverse problems, while traditional numerical methods, such as finite element methods [21] and finite difference methods [22], require well-defined initial and boundary conditions, which is unavailable in most applications. However, PINN is much slower than traditional numerical methods [23]. Therefore, how to combine the advantages of PINN and achieve numerical equivalent computational costs to quantification uncertainty is a critical challenge for solving PDE-based problems.

Recently, a novel NN called extreme learning machine (ELM) has received great attention owing to its low computational cost and simplicity [24, 25]. The most peculiar property of the ELM is that its input layer weights are pre-set random values and fixed throughout the training process, and output layer weights are training parameters. Spirited by ELM and PINN, Dwivedi et al. [26, 27] develop an ELM-based PDE solver called physic-informed extreme learning machine (PIELM) as a rapid version of PINN to solve the linear PDE problems efficiently. The PDE problem is transformed into a linear least squares problem by incorporating physics laws into the ELM as the cost function. Subsequently, the input layer weights are set randomly from a range according to the number of neurons, which is effective to solve elliptic PDEs with sharp gradients [28]. Besides, for nonlinear PDE problems, ELM-based PDE solvers transform the PDE problems into nonlinear least squares problems [29, 30], where the NN is also trained by a nonlinear least squares computation, not by gradients-based algorithms. The ELM-based PDE solver as a data-driven technology inevitably faces noise scenarios in real-life applications. The ELM-based PDE solver, however, has two evident drawbacks for ill-posed problems with noisy data:

1. For linear PDE problems, the output weights solved by Moore–Penrose generalized inverse easily suffer from overfitting noisy data and the accuracy of the solver is drastically sensitive to the number of hidden neurons under noise scenarios.
2. The ELM-based PDE solvers are not equipped with the built-in uncertainty quantification at current works, which may restrict their real applications.

In this paper, we propose a novel Bayesian physics-informed extreme learning machine (BPIELM) to solve both forward and inverse PDE problems with noisy data, see Fig.1, where the Bayesian method is used to quantify the uncertainty from the scattered noisy data. The Bayesian framework naturally quantifies the uncertainty arising from noisy data [31]. BPIELM consists of two parts: the prior for the extreme learning machine with physics laws and the Bayesian method for esti-

mating posterior distributions of parameters. The first part is to incorporate physics laws into the ELM as the cost function and introduce a prior distribution in the output layer weights, then the PDE problem is transformed into a linear least squares problem. In particular, for some inverse problems, the PDE problem parameters are considered as new output layer weights. The second part is to estimate posterior distributions of parameters and quantify the prediction uncertainty for the solution. We demonstrate the BPIELM method over a range of forward as well as inverse PDE problems and conduct a systematic comparison with PIELM and PINN. In summary, the followings are main benefits of BPIELM,

1. BPIELM is equipped with the built-in uncertainty quantification, which provides more accurate predictions than PIELM under noise scenarios and has lower computational costs than PINN.
2. Compared to PIELM, BPIELM avoids overfitting by estimating the probability distribution of output values instead of fitting noisy data, and hence is not drastically sensitive to the number of hidden neurons.
3. BPIELM is also used for some inverse problems in a unified framework, which achieves a competitive prediction accuracy with PIELM and PINN under noise scenarios.

The rest of the paper is structured as follows. Problem statement is presented in section 2. In section 3, we first briefly review PINN and PIELM, and then present the BPIELM method in detail. Numerical analysis is conducted in section 4, where we discuss the results of BPIELM, PIELM and PINN for forward and inverse PDE problems. The conclusions of the paper are in section 5.

## 2. Problem Setup

Suppose we have a linear partial differential equation (PDE) describing a physical system as follows:

$$\begin{aligned} Lu &= f_\lambda(x, y), (x, y) \in \Omega, \\ \mathcal{B}[u(x, y)] &= b(x, y), (x, y) \in \partial\Omega, \end{aligned} \tag{1}$$

where  $L$  is a linear differential operator,  $\mathcal{B}$  is a boundary/initial condition operator acting on the boundary  $\partial\Omega$ , while  $u(x, y)$  and  $f_\lambda(x, y)$  are prescribed the solution and source terms parameterized by the problem parameter  $\lambda$ , respectively.

In this work, we consider two types of data-driven problems, as summarized in Table 1 and delineated below (refer to [1] for more problems). The first scenario pertains to a forward PDE problem, where  $L$  and  $\mathcal{B}$  are known. The problem parameter  $\lambda$  is also known and the solution  $u(x, y)$  is unknown. Given noisy data of  $b$  sampled on  $\partial\Omega$ , our quantity of interest is to infer the solution  $u(x, y)$  at every  $(x, y) \in \Omega$  and quantify its uncertainty. Assume we only have  $N_b$  sensors on the boundary (including sensors of initial condition for simplification) and the dataset in this scenario is described as  $\mathcal{D} = \{\mathcal{D}_b\}$ ,  $\mathcal{D}_b = \left\{ (x_b^{(i)}, y_b^{(i)}), \bar{b}^{(i)} \right\}_{i=1}^{N_b}$ . Assume that the measurements are independently Gaussian distributed centered at the hidden real value as follows:

$$\bar{b}^{(i)} = b\left(x_b^{(i)}, y_b^{(i)}\right) + \epsilon_b^{(i)}, \quad i = 1, 2 \dots N_b, \quad (2)$$

where  $\epsilon_b^{(i)}$  is independent Gaussian noises with zero mean. Assume that the fidelity of each sensor is known and the standard deviation of  $\epsilon_b^{(i)}$  is known to be  $\sigma_b^{(i)}$ .

The second scenario pertains to an inverse PDE problem, where  $L$  and  $\mathcal{B}$  are known. The problem parameter  $\lambda$  is unknown, while  $u(x, y)$  is partially known and some extra measurements on  $u(x, y)$  are available. Given noisy data of  $u$  in  $\Omega$  as well as noisy data of  $b$ , our quantity of interest is to infer not only the solution  $u(x, y)$  at every  $(x, y) \in \Omega$  but the problem parameter  $\lambda$ , and quantify its uncertainty. Assume we have  $N_u$  sensors in the domain  $\Omega$  and  $N_b$  sensors on the boundary  $\partial\Omega$ . The dataset in this scenario is described as  $\mathcal{D} = \{\mathcal{D}_u, \mathcal{D}_b\}$ , where  $\mathcal{D}_u = \left\{ (x_u^{(i)}, y_u^{(i)}), \bar{u}^{(i)} \right\}_{i=1}^{N_u}$  and  $\mathcal{D}_b = \left\{ (x_b^{(i)}, y_b^{(i)}), \bar{b}^{(i)} \right\}_{i=1}^{N_b}$ . Similarly, the measurements from  $\mathcal{D}_u$  and  $\mathcal{D}_b$  are represented as

$$\begin{aligned} \bar{u}^{(i)} &= u\left(x_u^{(i)}, y_u^{(i)}\right) + \epsilon_u^{(i)}, \quad i = 1, 2 \dots N_u, \\ \bar{b}^{(i)} &= b\left(x_b^{(i)}, y_b^{(i)}\right) + \epsilon_b^{(i)}, \quad i = 1, 2 \dots N_b, \end{aligned} \quad (3)$$

where  $\epsilon_u^{(i)}$  is also the independent Gaussian noise with zero mean. Assume that the fidelity of each sensor is known and the standard deviation of  $\epsilon_u^{(i)}$  is known to be  $\sigma_u^{(i)}$ .

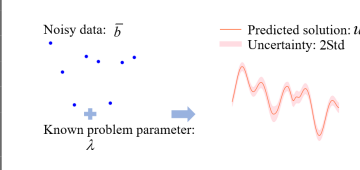
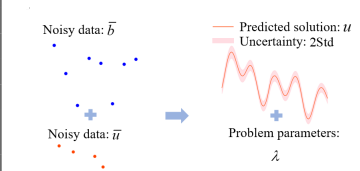
### 3. Method

#### 3.1. A primer in physics-informed neural network and physics-informed extreme learning machine

PIELM is considered as a rapid version of PINN, while BPIELM is the combination of PIELM and the Bayesian methods. Therefore, in this subsection, we briefly review PINN and PIELM.

**Table 1**

Problem considered in this paper.

Considered data-driven problem corresponding to following equation	
$Lu = f_\lambda(x, y), (x, y) \in \Omega$ , and $\mathcal{B}[u(x, y)] = b(x, y), (x, y) \in \partial\Omega$	
<b>Forward PDE problem</b>	
$L, \mathcal{B}$	known
$f_\lambda, b$	noisy data
$u(x, y)$	unknown
$\lambda$	known
Dataset	$\mathcal{D} = \{\mathcal{D}_b\}$ , where $\mathcal{D}_b = \left\{ (x_b^{(i)}, y_b^{(i)}, \bar{b}^{(i)}) \right\}_{i=1}^{N_b}$
Objective	$u(x, y)$ , uncertainty
	
<b>Inverse PDE problem</b>	
$L, \mathcal{B}$	known
$f_\lambda, b$	noisy data
$u(x, y)$	partially known
$\lambda$	unknown
Dataset	$\mathcal{D} = \{\mathcal{D}_u, \mathcal{D}_b\}$ , where $\mathcal{D}_u = \left\{ (x_u^{(i)}, y_u^{(i)}, \bar{u}_i) \right\}_{i=1}^{N_u}$ , $\mathcal{D}_b = \left\{ (x_b^{(i)}, y_b^{(i)}, \bar{b}_i) \right\}_{i=1}^{N_b}$
Objective	$\lambda, u(x, y)$ , uncertainty
	

### 3.1.1. Physics-informed neural network

PINN proposed by Raissi et al. [5] is utilized to solve forward and inverse PDE problems. The key idea of PINN is to encode the physics laws into loss function by adding a penalty term to constrain the space of admissible solutions and then the problem of solving PDEs in Eq.(1) is transformed into an optimization problem of minimizing the loss function. Concretely, a multi-layer perception with the activation function is first constructed. Then, to measure the difference between the NN and physics laws, the loss function is defined as

$$\mathcal{L}(\theta) = w_f \mathcal{L}_f(\theta; \mathcal{T}_f) + w_b \mathcal{L}_b(\theta; \mathcal{T}_b) + w_u \mathcal{L}_u(\theta; \mathcal{T}_u), \quad (4)$$

where

$$\begin{aligned} \mathcal{L}_f(\theta; \mathcal{T}_f) &= \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \|f(t, \mathbf{x})\|^2, \\ \mathcal{L}_b(\theta; \mathcal{T}_b) &= \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\mathcal{B}(\mathbf{x}) - b\|^2, \\ \mathcal{L}_u(\theta; \mathcal{T}_u) &= \frac{1}{|\mathcal{T}_u|} \sum_{\mathbf{x} \in \mathcal{T}_u} \|\hat{u}(\mathbf{x}) - u(\mathbf{x})\|^2. \end{aligned} \quad (5)$$

$w_f, w_b$  as well as  $w_u$  are hyper-parameters of weights.  $f$  represents the PDE residual in Eq.(1).  $\theta$  represents weights and bias of the NN.  $\mathcal{T}_f$ ,  $\mathcal{T}_b$ , and  $\mathcal{T}_u$  denote training points from PDE residual, boundary and data constraints, respectively. In other word, the loss function of PINN is a weighted sum of multiple terms, including PDE, boundary and data constraints.

Finally the NN is trained to search for the best parameters  $\theta$  by minimizing the loss function, where automatic differentiation is used to minimize the loss function.

### 3.1.2. Physics-informed extreme learning machine

Spirited by PINN, Dwivedi et al. [26] propose a rapid version of PINN called PIELM, which can be applied to stationary and time-dependent linear PDEs. The most peculiar property is that PIELM is a single-layer feed-forward NN, where its input layer weights are fixed with random values and only output layer weights are training parameters. Then, the physics laws are incorporated into the NN as the cost function. In particular, the linear nature of linear PDEs and the physics laws in Eq.(1) are combined to solve a linear equation as follows:

$$\mathbf{H}\mathbf{c} = \mathbf{T}, \quad (6)$$

where  $\mathbf{c}$  is the unknown output layer weights and  $\mathbf{H}$  as well as  $\mathbf{T}$  are known matrices that contain physic laws (PDE, boundary and data constraints). Finally, the predictions of  $\mathbf{c}$  come from linear least squares solution by

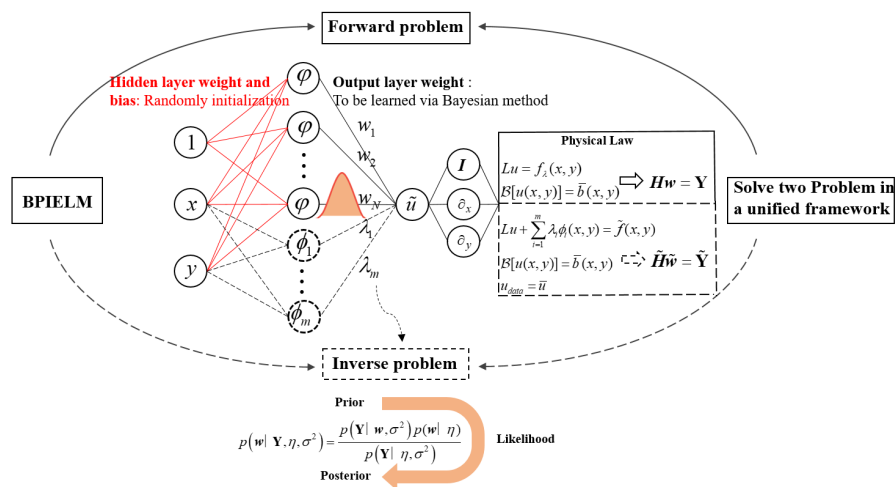
$$\mathbf{c}_{pred} = \mathbf{H}^\dagger \mathbf{T}, \quad (7)$$

where  $\mathbf{H}^\dagger = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$  represents the Moore–Penrose generalized inverse matrix of  $\mathbf{H}$ .

### 3.2. Bayesian physics-informed extreme learning machine

BPIELM is an ELM-based PDE solver that combines two ideas from PIELM and the Bayesian method. The first idea, coming from PIELM, is to incorporate the physics laws into ELM as the cost function, making linear PDE problems transformed into a linear least squares problem. The other one, coming from Bayesian ELM [32, 33], is combining the Bayesian method with ELM to naturally quantify the uncertainty for regression and classification tasks. The output layer parameters are defined as a prior distribution and are optimized by the Bayesian method, not by the Moore–Penrose generalized inverse. Since PIELM is not equipped with the built-in uncertainty quantification and is

prone to overfit noisy data, we resolve two drawbacks by combining characteristics between PIELM and the Bayesian framework. In addition, we also unify the forward and inverse PDE problems in the one framework. BPIELM consists of two parts: the prior for extreme learning machine with physics laws and the Bayesian method for estimating posterior distributions of parameters. The first part is incorporating physics laws into the NN as the cost function and introducing a prior in the output layer weights, then the PDE problem is transformed into a linear least squares problem. The second part is to estimate posterior distributions of parameters and quantify the prediction uncertainty. Fig.1 shows the conceptual flow of the BPIELM method and the detailed descriptions are provided in the following sections.



**Fig. 1.** Conceptual flow of the BPIELM method with noisy data.  $p(\omega | \eta)$  is the prior for the output layer weights as well as problem parameters,  $p(\mathbf{Y} | \omega, \sigma^2)$  represents the likelihood of measurements, and  $p(\omega | \mathbf{Y}, \eta, \sigma^2)$  represents the posterior.

### 3.2.1. The prior for extreme learning machine with physics laws

We consider a single-layer NN with  $N$  neurons in Fig.1. The key of the ELM is to pre-set the input layer weights randomly, which are fixed throughout the training process. The parameters of the input layer are defined as  $\chi = [x, y, 1]^T$ ,  $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_n]^T$ ,  $\beta = [\beta_1, \beta_2, \dots, \beta_n]^T$ ,  $\gamma = [\gamma_1, \gamma_2, \dots, \gamma_n]^T$ , where  $\alpha$ ,  $\beta$  and  $\gamma$  are input layer weights. The nonlinear activation function is  $\varphi = \tanh$  and the output layer weights are  $\omega = [\omega_1, \omega_2, \dots, \omega_n]^T$ . Then the output of  $k^{th}$  hidden



neuron is represented by  $\varphi(z_k)$ , where  $z_k = [\alpha_k, \beta_k, \gamma_k]\boldsymbol{\chi}$ . The output of the NN is written as

$$\hat{u}(\boldsymbol{\chi}) = \varphi(\mathbf{z})\boldsymbol{\omega}. \quad (8)$$

Given  $\hat{u}(\boldsymbol{\chi})$ , the partial derivatives of Eq.(8), with respect to the independent variables, are

$$\begin{aligned} \frac{\partial^k \hat{u}}{\partial x^k} &= \sum_{j=1}^n \omega_j \alpha_j^k \frac{\partial^k \varphi}{\partial z^k}, \\ \frac{\partial^k \hat{u}}{\partial y^k} &= \sum_{j=1}^n \omega_j \beta_j^k \frac{\partial^k \varphi}{\partial z^k}, \end{aligned} \quad (9)$$

where  $k$  refers to the  $k^{\text{th}}$  order derivative.

Therefore, according to Eq.(1), (2) and (3), the physics laws are incorporated into the NN by introducing physics-based training errors,  $\boldsymbol{\xi}_f, \boldsymbol{\xi}_b$ . The training errors  $\boldsymbol{\xi}_f, \boldsymbol{\xi}_b$  are defined as errors in approximating PDE and boundary/initial condition, respectively,

$$\begin{aligned} \boldsymbol{\xi}_f &= L\varphi(\mathbf{z})\boldsymbol{\omega} - f_\lambda(x, y) = 0, \\ \boldsymbol{\xi}_b &= \mathcal{B}[\varphi(\mathbf{z})\boldsymbol{\omega}] - \bar{b}(x, y) = 0. \end{aligned} \quad (10)$$

For the forward problems, Eq.(10) leads to a linear equation, which is represented as

$$\mathbf{H}\boldsymbol{\omega} = \mathbf{Y}, \quad (11)$$

where  $\mathbf{H} \in \mathbb{R}^{N^* \times N}$ ,  $N^* = N_f + N_b$ ,  $N$  is equal to the neuron number.  $\mathbf{H}$  and  $\mathbf{Y}$  are matrices determined by  $L$  and  $\mathcal{B}$ . The output layer weight  $\boldsymbol{\omega}$  is the training parameters.

In particular, for some inverse PDE problems in this work, we consider the type of PDEs and its training errors in approximating the PDE can be written by separating variables as

$$\boldsymbol{\xi}_f = L\varphi(\mathbf{z})\boldsymbol{\omega} + \sum_{j=1}^m \phi_j \lambda_j - \tilde{f} = 0, \quad (12)$$

where  $\phi_j$  and  $\tilde{f}$  are determined by separating variables. Under noisy data in Eq.(3), the training

errors in approximating the boundary/initial condition and data constraints are represented by

$$\begin{aligned} \boldsymbol{\xi}_b &= (H_b, 0) \begin{pmatrix} \boldsymbol{\omega}, \\ \boldsymbol{\lambda}, \end{pmatrix} - \bar{b}(x, y) = 0, \\ \boldsymbol{\xi}_u &= (H_u, 0) \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{\lambda} \end{pmatrix} - \bar{u}(x, y) = 0. \end{aligned} \tag{13}$$

For the inverse problems, Eq.(12) and (13) also lead to a linear system,

$$\tilde{\mathbf{H}}\tilde{\boldsymbol{\omega}} = \tilde{\mathbf{Y}}, \tag{14}$$

where  $\tilde{\boldsymbol{\omega}} = [\boldsymbol{\omega}, \boldsymbol{\lambda}]^T$ ,  $\tilde{\mathbf{H}} \in \mathbb{R}^{N^* \times N}$ ,  $N^* = N_f + N_b + N_u$  as well as  $\tilde{\mathbf{Y}}$ , are matrices determined by the  $L$  and  $\mathcal{B}$ . The problem parameters are also considered as output layer weights.

In summary, physics laws are encoded into the linear system by using the linear nature of linear PDEs. To simplify the following representation, the linear system for forward or inverse problems is written as  $\mathbf{H}\boldsymbol{\omega} = \mathbf{Y}$ , where  $\boldsymbol{\omega}$  is the training parameters. In most applications of Bayesian theory, a commonly used prior for  $\boldsymbol{\omega}$  is that each component of  $\boldsymbol{\omega}$  is an independent Gaussian distribution with zero mean, which is represented by

$$p(\boldsymbol{\omega} \mid \eta) = \mathcal{N}(\mathbf{0}; \eta^{-1}\mathbf{I}), \tag{15}$$

where  $\mathbf{I}$  is the identity matrix and  $\eta$  is a hyperparameter, which will be optimized the following steps.

### 3.2.2. The Bayesian method for estimating posterior distributions of parameters

In this subsection, the Bayesian method is used to estimate posterior distributions of parameters, which improves the over-fitting problem of ELM-based solvers and quantifies the prediction uncertainty. According to wide application of Bayesian theory, most models are divided into two steps:

1. Inference of the posterior distribution of model parameters. Suppose  $\boldsymbol{\omega}$  is the set of parameters and  $\mathcal{D}$  is the dataset. The posterior distribution is proportional to the product of the prior

distribution and the likelihood function,

$$P(\boldsymbol{\omega} | D) \propto P(\boldsymbol{\omega})P(D | \boldsymbol{\omega}). \quad (16)$$

2. The output  $\mathbf{Y}_{new}$  can be given by the integral of the posterior distribution of  $\boldsymbol{\omega}$  for input instances of  $\boldsymbol{\chi}_{new}$ ,

$$P(\mathbf{Y}_{new} | \boldsymbol{\chi}_{new}, D) = \int P(\mathbf{Y}_{new} | \boldsymbol{\chi}_{new}, \boldsymbol{\omega}) P(\boldsymbol{\omega} | D) d\boldsymbol{\omega}. \quad (17)$$

Suppose the linear system follows the relationship,

$$\mathbf{Y} = \mathbf{H}\boldsymbol{\omega} + \epsilon, \quad (18)$$

where  $\epsilon$  follows a Gaussian distribution  $\mathcal{N}(0; \sigma^2)$ . The probability model and likelihood function are then derived as

$$P(\mathbf{Y} | \boldsymbol{\omega}, \sigma^2) = \left( \frac{1}{2\pi\sigma^2} \right)^{\frac{N^*}{2}} \exp\left( -\frac{\|\mathbf{Y} - \mathbf{H}\boldsymbol{\omega}\|^2}{\sigma^2} \right), \quad (19)$$

where  $N^*$  is the dimension number of  $\mathbf{H}$ .

Under the assumption that the prior distribution and the likelihood function follow Gaussian distributions, the posterior can be derived as

$$p(\boldsymbol{\omega} | \mathbf{Y}, \eta, \sigma^2) = \frac{p(\mathbf{Y} | \boldsymbol{\omega}, \sigma^2) p(\boldsymbol{\omega} | \eta)}{p(\mathbf{Y} | \eta, \sigma^2)}, \quad (20)$$

which also follows a Gaussian distribution with a mean value of  $\boldsymbol{\mu}$  and a variance of  $\boldsymbol{\Sigma}$  [34],

$$\begin{aligned} \boldsymbol{\mu} &= \sigma^{-2} \boldsymbol{\Sigma} \mathbf{H}^T \mathbf{Y}, \\ \boldsymbol{\Sigma} &= \left( \eta \mathbf{I} + \sigma^{-2} \mathbf{H}^T \mathbf{H} \right)^{-1}. \end{aligned} \quad (21)$$

Then the optimal values of hyper-parameters,  $\eta$  and  $\sigma$ , are obtained by Evidence Procedure

[35]. The optimal conditions are iteratively calculated by

$$\begin{aligned}\eta &\leftarrow \frac{N-\eta \cdot \text{trace}[\boldsymbol{\mu}]}{\boldsymbol{\Sigma}^T \boldsymbol{\Sigma}}, \\ \sigma^2 &\leftarrow \frac{\|\mathbf{Y}-\mathbf{H}\boldsymbol{\omega}\|^2}{N^*-N+\eta \cdot \text{trace}[\boldsymbol{\mu}]},\end{aligned}\tag{22}$$

where  $N^*$  is the dimension number of  $\mathbf{H}$  and  $N$  is the number of parameters. The iterative process stops when  $\boldsymbol{\mu}$  falls below a given threshold.

Finally, given a  $\boldsymbol{\chi}_{\text{new}}$ , the posterior distributions of the parameters are used by Eq.(17) to obtain the output  $\mathbf{Y}_{\text{new}}$ . Besides, the output follows a distribution,

$$p(\mathbf{Y}_{\text{new}} | \mathbf{Y}, \eta, \sigma^2) = \mathcal{N}(\mathbf{H}_{\text{new}}\boldsymbol{\omega}; \sigma^2(\boldsymbol{\chi}_{\text{new}})),\tag{23}$$

where  $\mathbf{H}_{\text{new}} = \varphi([\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}] \boldsymbol{\chi}_{\text{new}})$  and the variance is calculated by

$$\sigma^2(\boldsymbol{\chi}_{\text{new}}) = \sigma^2 + \boldsymbol{\chi}_{\text{new}}^T \boldsymbol{\Sigma} \boldsymbol{\chi}_{\text{new}}.\tag{24}$$

#### 4. Numerical result

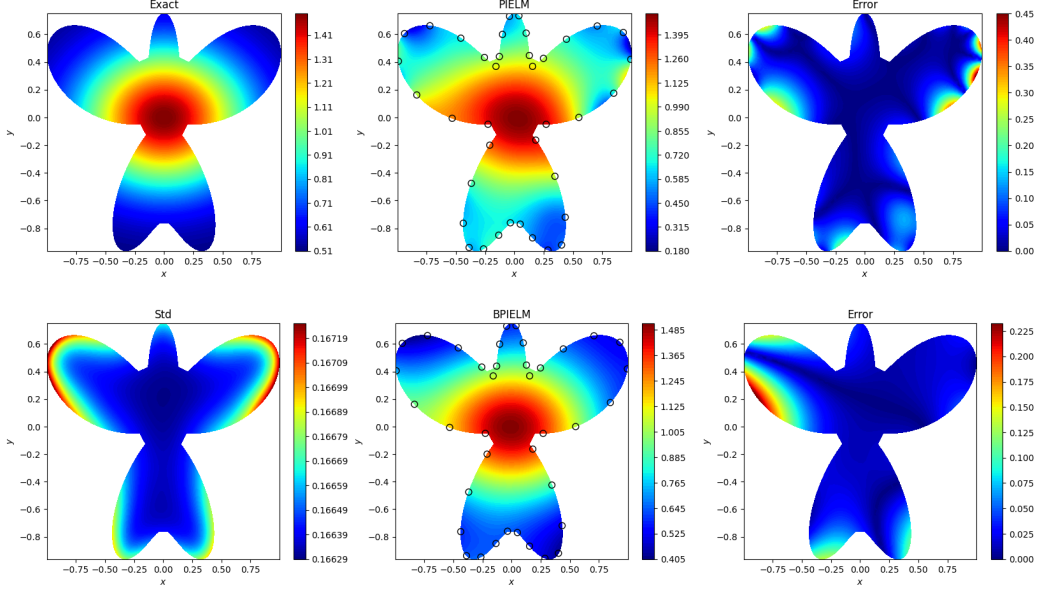
In this section, we will first consider solving forward and inverse PDE problems by BPIELM over PIELM and PINN. The forward PDE problem including Poisson, advection and diffusion equations. The inverse PDE problem includes Poisson and Helmholtz equations, where noisy data is used to solve the solution and identify the unknown problem parameters. All the experiments are conducted the very common machine with a 2.8-GHz Intel(R) Xeon(R) Gold 6242 CPU, 128-GB memory, and NVIDIA GeForce RTX 3090 GPU.

##### 4.1. Poisson equation

The two dimensional (2D) Poisson equation is given by

$$u_{xx} + u_{yy} = (16x^2 + 64y^2 - 12) e^{-(2x^2+4y^2)}, (x, y) \in \Omega,\tag{25}$$

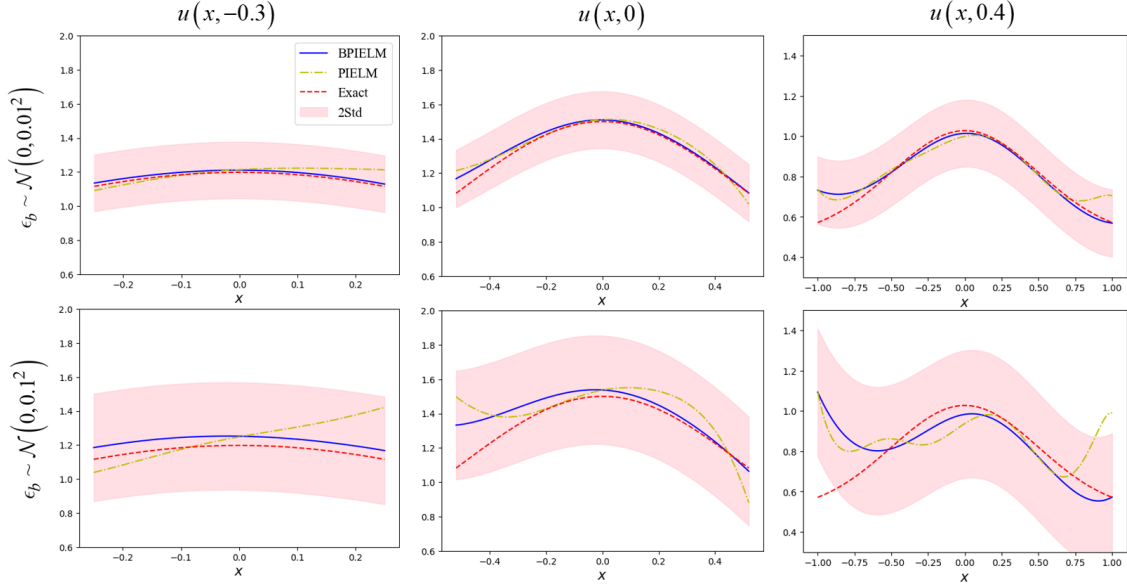
where the solution is  $u = \frac{1}{2} + e^{-(2x^2+4y^2)}$ ,  $\Omega = \{(x, y) | x = 0.55\rho(\theta) \cos(\theta), y = 0.75\rho(\theta) \sin(\theta)\}$ , and  $\rho(\theta) = 1 + \cos(\theta) \sin(4\theta), 0 \leq \theta \leq 2\pi$ . Assuming that the boundary condition is unknown, we only have  $N_b$  sensors on the boundary, which are equidistantly distributed on  $\partial\Omega$ , and consider the Gaussian noise in the measurements.



**Fig. 2.** Poisson equation: The first row represents the exact solution, the PIELM solution and its absolute error, respectively. The second row represents the standard deviations for  $u$  using BPIELM, the BPIELM solution and its absolute error, respectively. The noise scale on the boundary is  $\epsilon_b \sim \mathcal{N}(0, 0.01^2)$ , the number of neurons is  $N = 100$ , and the number of training points is  $N_f = 400$  and  $N_b = 38$ . Black circles represent the positions of sensors.

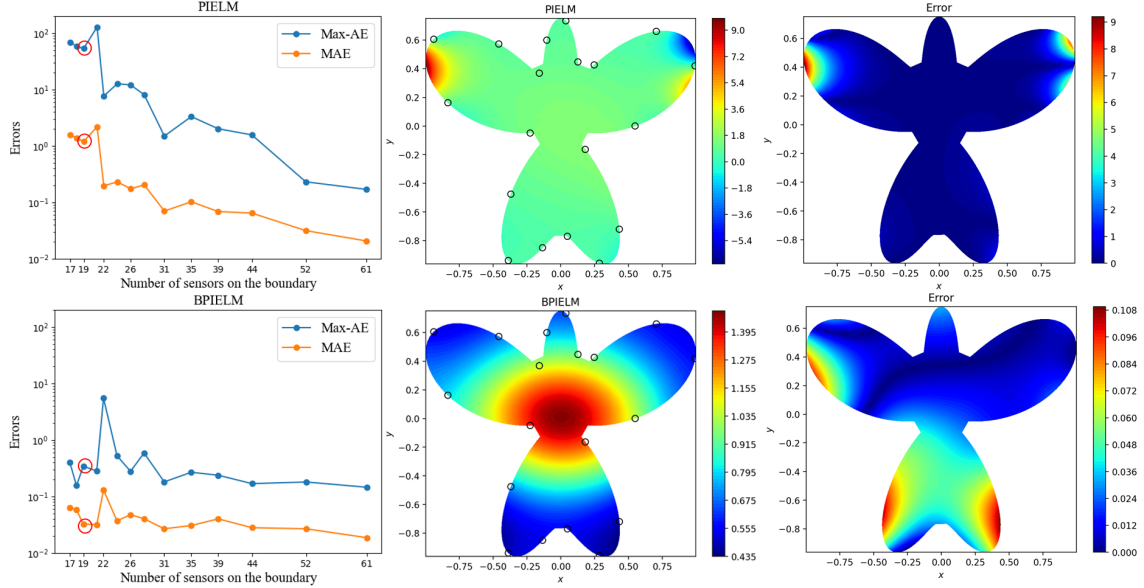
To solve the Poisson equation in the BPIELM framework, we encode governing equations in Eq.(25) and sensor measurements on the boundary into the linear system. Then, we use the Bayesian method to estimate the output layer weights, where the hyperparameters are  $\eta = 2e - 1$  and  $\beta = 1$  (the default setting in the following tests). In this case, the number of neurons is  $N = 100$  and the number of training points is  $N_f = 400$  and  $N_b = 38$ . The noise scale on the boundary is  $\epsilon_b \sim \mathcal{N}(0, 0.01^2)$ . Fig.2 shows the results of PIELM and BPIELM, where BPIELM provides more accurate predictions for the solution and gives the uncertainty quantification for the solution. In BPIELM, the standard deviation for the case is large on the boundary even though there are some sensors on the boundary. Besides, the area with a large error has a relatively large standard deviation. The mean absolute errors (MAEs) of PIELM and BPIELM are 0.029 and 0.019, respectively. The areas with large errors are mainly concentrated in the areas without sensors, while BPIELM gives predicted means closer to the exact solutions on the right-wing of the butterfly than PIELM. Fig.3 provides further comparisons between PIELM and BPIELM under two different noise scales, i.e.,  $\epsilon_b \sim \mathcal{N}(0, 0.01^2)$  and  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . BPIELM provides predictions

closer to the exact solution than PIELM under two noise scales, and the errors of BPIELM are mostly bounded by two standard deviations, which increase as the noise scale increases. Moreover, the errors between predictions and the exact solution increase with the increasing noise scale.

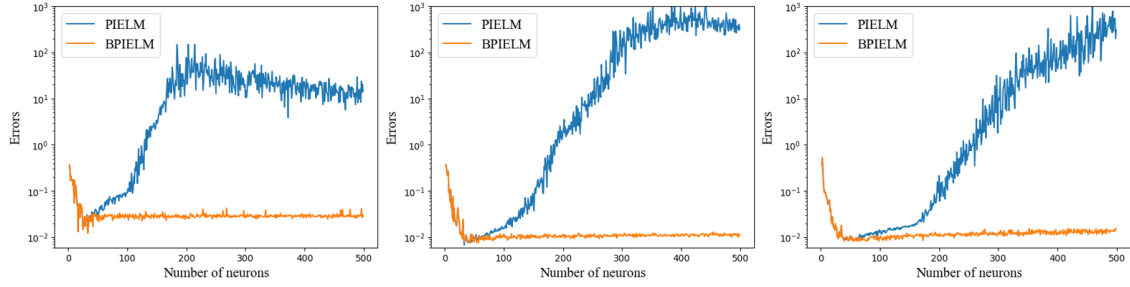


**Fig. 3.** Poisson equation: The columns (from left to right) represent comparisons between BPIELM and PIELM at  $y = -0.3$ ,  $y = 0$  and  $y = 0.4$ , where  $N = 100$ ,  $N_f = 400$  and  $N_b = 41$  are used. The row (from top to down) represents two noise scales on the boundary,  $\epsilon_b \sim \mathcal{N}(0, 0.01^2)$  and  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ .

The effect of the different number of sensors on the boundary for PIELM and BPIELM is illustrated in Fig.4. In this test, the number of neurons is  $N = 100$ , the number of training points is  $N_f = 200$ , and the noise scale is  $\epsilon_b \sim \mathcal{N}(0, 0.01^2)$ . In the first column, we compare the maximum absolute error on the whole domain (Max-AE) and the MAE of PIELM and BPIELM. BPIELM achieves much better MAEs than PIELM with a smaller number of sensors. Besides, BPIELM achieves smaller Max-AEs than PIELM at the same number of sensors. This is because the pseudo-inverse operation is used in PIELM so as to be prone to overfit noisy data (sensor measurements on the boundary), causing the abnormal predictions at some points, but BPIELM alleviates overfitting to decrease the Max-AE. Especially, the results of PIELM and BPIELM with 19 sensors are shown in the last two columns. The errors of PIELM on the two wings of the butterfly tend to exceed 9, but BPIELM reduces the error to be less than 0.1. In summary, the above results show that the number of sensors has a stronger effect for PIELM on the predictive accuracy than BPIELM.



**Fig. 4.** Poisson equation: The first column represents the Max-AE and the MAE under different number of sensors on the boundary, where  $\epsilon_b \sim \mathcal{N}(0, 0.01^2)$ ,  $N = 100$  and  $N_f = 200$  are used. The second and third columns represent the predictions at  $N_b = 19$  and its absolute error. The rows (from top to down) represent the results of PIELM and BPIELM. Black circles represent the positions of sensors.



**Fig. 5.** Poisson equation: The columns (from left to right) represent the MAE comparisons between BPIELM and PIELM at  $N_{bc} = 21$ ,  $N_{bc} = 42$  and  $N_{bc} = 63$ , where  $\epsilon_b \sim \mathcal{N}(0, 0.01^2)$  and  $N_f = 400$  are used.

PIELM is unstable for the neuron number and is easily affected by noisy data. To this end, we compare the effect of the different neuron number for a fixed number of training points in PIELM and BPIELM on the predictive accuracy. The results are shown in Fig.5. Under noise scenarios, PIELM hardly achieves the best MAE when the number of neurons is equal to or close to the number of constraints, i.e.,  $N = N_f + N_b$ . The solution accuracy of PIELM is sensitive to the number of neurons. Under the small neuron number, MAEs between PIELM and BPIELM

**Table 2**

Poisson equation: The comparisons of PINN, PIELM and BPIELM under three noise scales, i.e.,  $\epsilon_b \sim \mathcal{N}(0, 0.01^2)$ ,  $\epsilon_b \sim \mathcal{N}(0, 0.05^2)$  and  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . The number of training points is  $N_f = 400$ ,  $N_b = 19$  and the number of neurons is  $N = 100$ . PINN has four hidden layers with 100 neurons in each layer and the epoch is 3,000.

Noise scale	0.01			0.05			0.1		
	Max-AE	MAE	Time/s	Max-AE	MAE	Time/s	Max-AE	MAE	Time/s
PINN	0.165	0.043	142.30	0.251	0.055	141.02	0.327	0.072	143.32
PIELM	0.447	0.029	1.21	0.998	0.065	1.22	1.41	0.092	1.22
BPIELM	0.232	0.019	2.05	0.516	0.047	2.67	0.759	0.067	2.65

are close. It means that a shallow NN architecture both for PIELM and BPIELM can not learn enough physics information to depict the whole solution. As the neuron number increases, MAEs of PIELM increase after reaching the minimum value in a certain range, while MAEs of BPIELM converge to a small value and fluctuate in a small range. It indicates that an unusually large neuron number in PIELM causes overfitting noisy data, but BPIELM alleviates the overfitting. Especially for  $N > N^*$ , BPIELM gives much better predictions for the solution than PIELM.

Table 2 is a further comparison of PINN, PIELM and BPIELM under three noise scales, i.e.,  $\epsilon_b \sim \mathcal{N}(0, 0.01^2)$ ,  $\epsilon_b \sim \mathcal{N}(0, 0.05^2)$  and  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . In terms of the computational cost (the network training time), PINN takes around 140 seconds to train with Adam [36]. In contrast, the training times of PIELM and BPIELM are on the order of a second and on the order of two seconds, respectively. For the accuracy of the solution, BPIELM provides more accurate predictions than PIELM. The MAE of BPIELM is better than PINN, while PINN is superior to BPIELM in terms of the Max-AE. In a word, BPIELM is not only much more accurate than PIELM but also considerably cheaper than PINN in terms of the computational cost.

#### 4.2. Advection equation

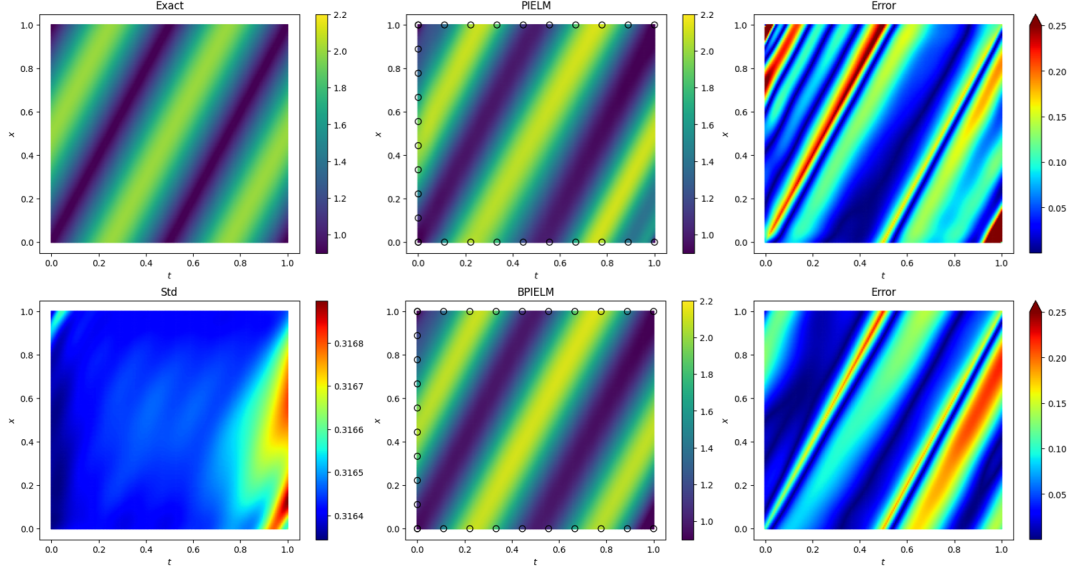
The advection equation in one and two spatial dimensions (plus time) is given by

$$\begin{aligned} \frac{\partial u}{\partial t} - c \frac{\partial u}{\partial x} &= 0, (x, t) \in \Omega, \\ u(x_1, t) &= u(x_2, t), u(x, 0) = h(x), \end{aligned} \tag{26}$$

where  $u(x, t)$  is the solution, the constant  $c = -2$  is the wave speed,  $x_1 = 0$  as well as  $x_2 = 1$  are the boundary points, and  $\Omega = \{(x, t) \mid x \in [0, 1], t \in [0, 1]\}$ . The exact solution is considered as

$$u(x, t) = 2 \operatorname{sech} \left[ 3 \left( -\frac{1}{2} + \xi \right) \right], \tag{27}$$





**Fig. 6.** Advection equation: The first row represents the exact solution, the PIELM solution and its absolute error, respectively. The second row represents standard deviations for  $u$  using BPIELM, the BPIELM solution and its absolute error. The data noise scale on the boundary is  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ , the number of neurons is  $N = 150$  and the number of training points is  $N_f = 400$  and  $N_b = 28$ . Black circles represent the positions of sensors.

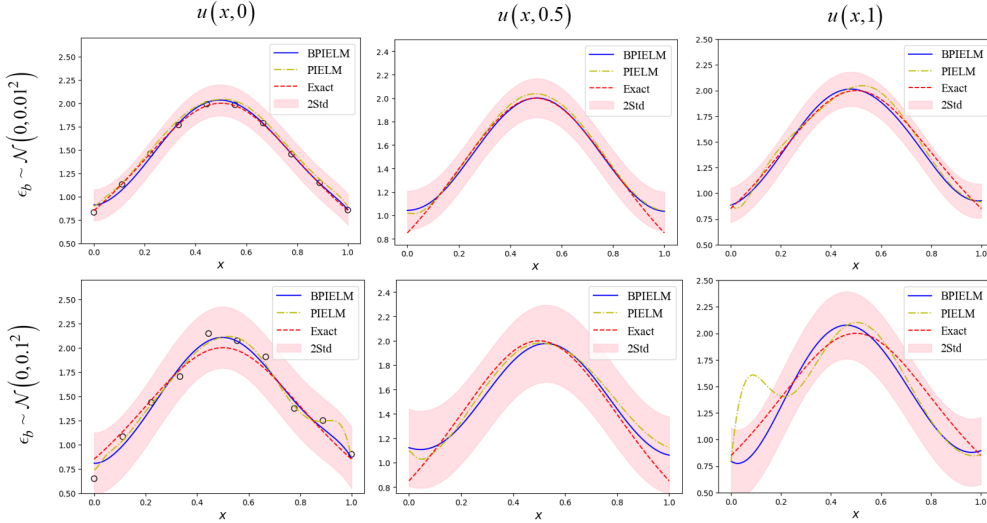
where

$$\xi = \text{mod} \left( x - x_0 + ct + \frac{L}{2}, L \right), \quad L = x_2 - x_1, \quad x_0 = 0.5. \quad (28)$$

Assuming that the boundary conditions ( $u(x_1, t)$ ,  $u(x_2, t)$  and  $h(x)$ ) are unknown, we only have  $N_b$  sensors on the boundary, which are equidistantly distributed on  $\partial\Omega$ , and consider the Gaussian noise in the measurements.

To employ BPIELM for solving the advection equation, we encode the governing equations and sensor measurements on the boundary into the linear system, and then use the Bayesian method to estimate the output layer weights. In this simulation, the number of neurons is  $N = 150$  and the number of training points is  $N_f = 400$  and  $N_b = 28$ . The noise scale on the boundary is  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . The results of PIELM and BPIELM are illustrated in Fig.6, where the data noise scale on the boundary is  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . Under the large noise scale, BPIELM provides better predictions for the solution than PIELM and gives the uncertainty quantification for the solution. Concretely, MAEs of PIELM and BPIELM are 0.070 and 0.066, respectively. For BPIELM, the area with large errors corresponds to relatively large standard deviations. Furthermore, according

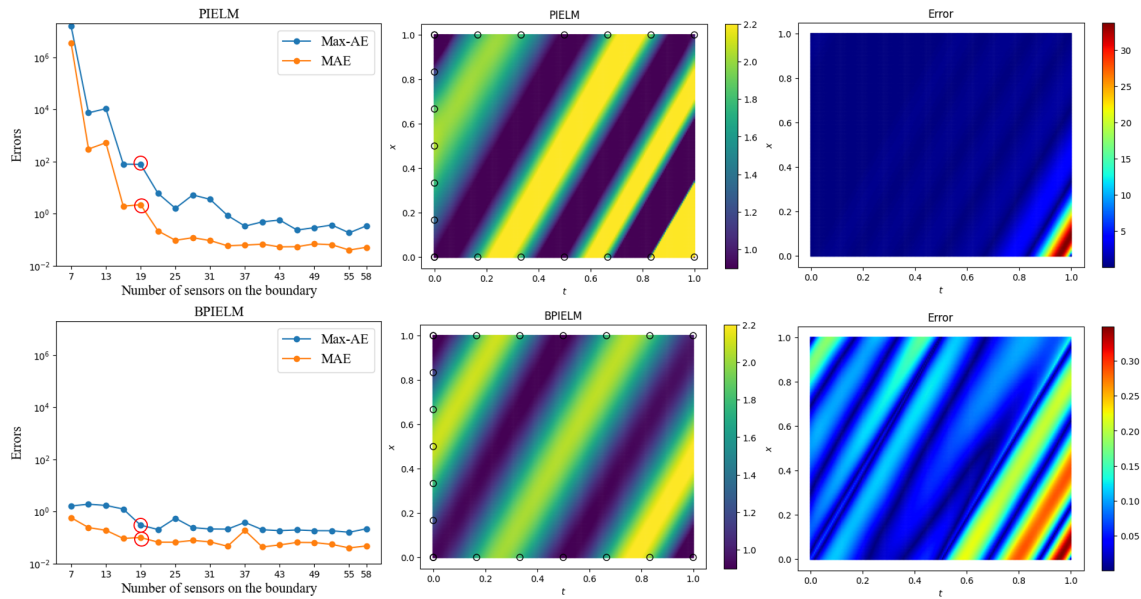
to three temporal snapshots  $t = 0, 0.5, 1$ , Fig.7 provides the comparisons between PIELM and BPIELM under two different noise scales, i.e.,  $\epsilon_b \sim \mathcal{N}(0, 0.01^2)$  and  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . The errors of BPIELM are mostly bounded by two standard deviations, and the errors between predictions and the exact solution increase with the increasing noise scale. Under the small noise scale, BPIELM and PIELM provide predictions close to the exact solution. When the noise scale is large, BPIELM gives much better predictions than PIELM, especially for the area near the boundary.



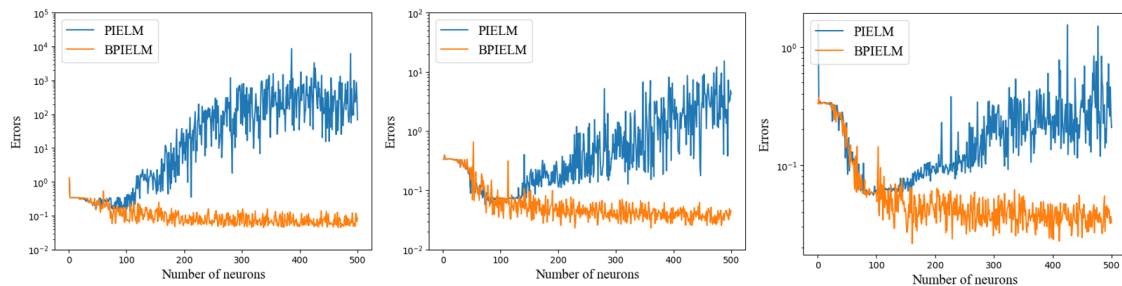
**Fig. 7.** Advection equation: The columns (from left to right) represent comparisons between BPIELM and PIELM at  $t = 0$ ,  $t = 0.1$  and  $t = 1$ , where  $N = 150$ ,  $N_f = 400$  and  $N_b = 28$  are used. The rows (from top to down) represent two noise scales on the boundary,  $\epsilon_b \sim \mathcal{N}(0, 0.01^2)$  and  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ .

Fig.8 illustrates the effect of the different number of sensors on the boundary for PIELM and BPIELM. In this test, the number of neurons is  $N = 150$ , the number of training points is  $N_f = 400$ , and the noise scale is  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . In the first column, we compare Max-AEs and MAEs of PIELM and BPIELM. Max-AEs and MAEs of PIELM are much worse than BPIELM at the same number of sensors, especially for the scenario with the small number of sensors. In particular, the last two columns show the predictions of PIELM and BPIELM with 19 sensors. The Max-AE of PIELM is extremely large, but BPIELM reduces the Max-AE to be less than 0.4. In a word, BPIELM outperforms PIELM at the same number of sensors in terms of Max-AEs and MAEs.

To further study the impact of the different neuron number on PIELM and BPIELM, we choose three scenarios with  $N_{bc} = 16, 28, 37$ . The results are shown in Fig.9. Under the noise scenario,



**Fig. 8.** Advection equation: The first column represents the maximum error and MAE under different number of sensors on the boundary, where  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ ,  $N = 150$  and  $N_f = 400$  are used. The second and third columns represent the prediction at  $N_b = 19$  and its absolute error. The rows (from top to down) represent the results of PIELM and BPIELM. Black circles represent the positions of sensors.



**Fig. 9.** Advection equation: The columns (from left to right) represent comparisons of MAEs between BPIELM and PIELM at  $N_{bc} = 16$ ,  $N_{bc} = 28$  and  $N_{bc} = 37$ , where  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$  and  $N_f = 400$  are used.

PIELM hardly achieves the best MAE when the number of neurons is equal to or close to the number of constraints, i.e.,  $N = N_f + N_b$ . Obviously, the solution accuracy of PIELM is sensitive to the number of neurons, while BPIELM gives much better predictions when the neuron number is larger enough. Moreover, as shown in the figure (from left to right), when the number of noisy measurements increases, MAEs of PIELM and BPIELM decreases and the fluctuation amplitude of the curve increases. However, the fluctuation amplitude of the curve using BPIELM is much

smaller than that using PIELM, which shows that BPIELM alleviates overfitting noisy data.

Table 3 presents further comparisons of PINN, PIELM and BPIELM with regard to the accuracy and computational cost under three noise scales. In terms of MAEs, BPIELM is much more accurate than PIELM and PINN, while Max-AEs of BPIELM are close to PINN. For PINN, PIELM and BPIELM, Max-AEs and MAEs increase as the noise increase. Besides, PIELM and BPIELM are dramatically faster to train PINN (by two orders of magnitude).

**Table 3**

Advection equation: The comparisons of PINN, PIELM and BPIELM under three noise scales. The number of training points is  $N_f = 400$ ,  $N_b = 28$  and the number of neurons is  $N = 150$ . PINN has four hidden layers with 100 neurons in each layer and the epoch is 4,000.

Noise scale	0.01			0.05			0.1		
	Max-AE	MAE	Time/s	Max-AE	MAE	Time/s	Max-AE	MAE	Time/s
PINN	0.162	0.033	153	0.183	0.055	151	0.223	0.082	153
PIELM	0.442	0.037	0.98	0.541	0.047	1.01	0.670	0.070	1.22
BPIELM	0.170	0.027	1.12	0.209	0.039	1.18	0.224	0.066	1.30

#### 4.3. Diffusion equation

The diffusion equation in one and two spatial dimensions (plus time) is given by

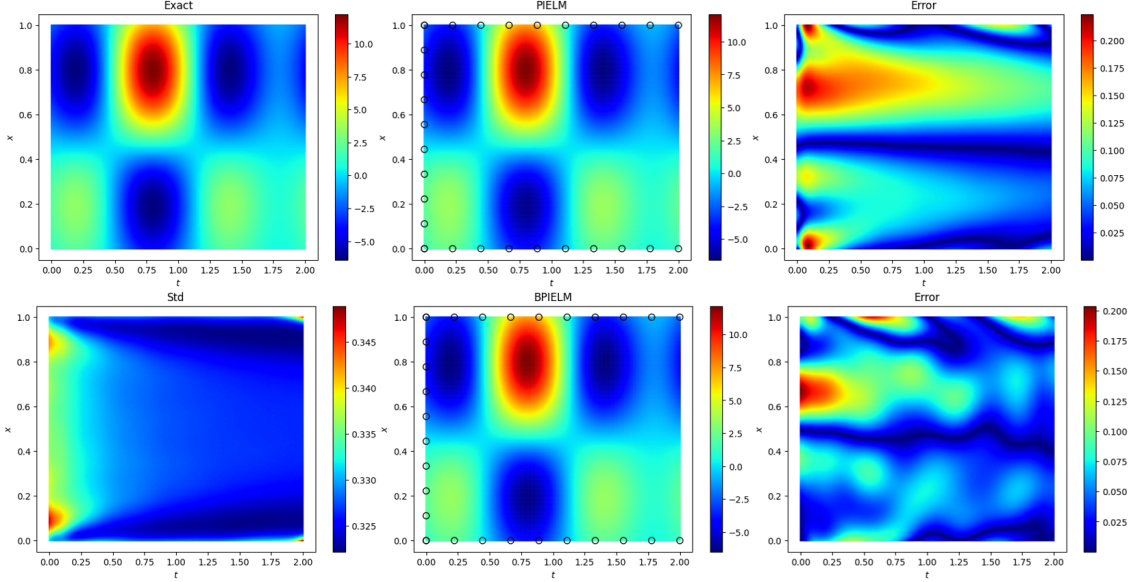
$$\begin{aligned}
 \frac{\partial u}{\partial t} - v \frac{\partial^2 u}{\partial x^2} &= f(x, t), (x, t) \in \Omega, \\
 u(x_1, t) &= b_1(t), u(x_2, t) = b_2(t), \\
 u(x, 0) &= h(x),
 \end{aligned} \tag{29}$$

where  $f(x, t)$  is a source term, the constant  $v = 0.01$  is the diffusion coefficient, and  $\Omega = \{(x, t) \mid x \in [0, 1], t \in [0, 2]\}$ . The expression for  $f(x, t)$  is constructed by choosing the following the exact solution,

$$u(x, t) = \left[ 2 \cos\left(\pi x + \frac{\pi}{5}\right) + \frac{3}{2} \cos\left(2\pi x - \frac{3\pi}{5}\right) \right] \left[ 2 \cos\left(\pi t + \frac{\pi}{5}\right) + \frac{3}{2} \cos\left(2\pi t - \frac{3\pi}{5}\right) \right]. \tag{30}$$

Assuming that the boundary conditions ( $b_1(t)$ ,  $b_2(t)$  and  $h(x)$ ) are unknown, we only have  $N_b$  sensors on the boundary, which are equidistantly distributed on  $\partial\Omega$ , and consider the Gaussian noise in the measurements.

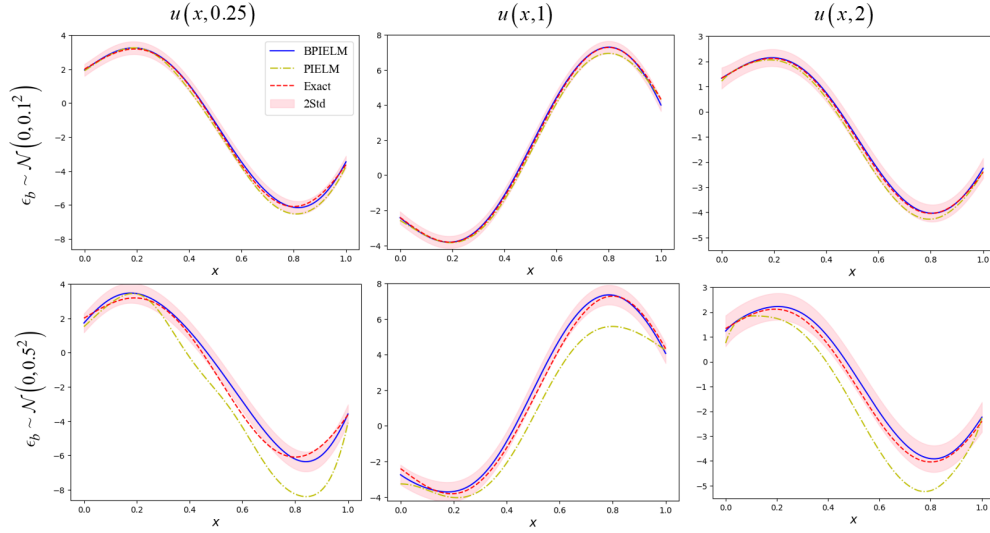
We approximate the solution using BPIELM to solve the diffusion equation. Concretely, the governing equations and the sensor measurements on the boundary are encoded into the linear



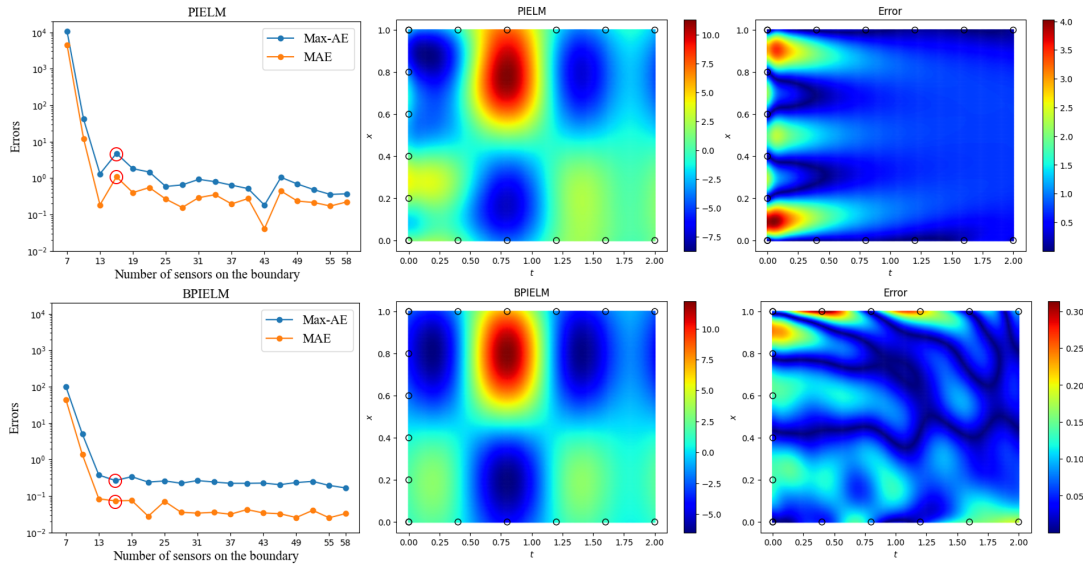
**Fig. 10.** Diffusion equation: The first row represents the exact solution, the PIELM solution and its absolute error, respectively. The second row represents the standard deviations for  $u$  using BPIELM, the BPIELM solution and its absolute error, respectively. The data noise scales on the boundary is  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ , the number of neurons is  $N = 180$  and the number of training points is  $N_f = 400$  and  $N_b = 28$ . Black circles represent positions of sensors.

system, and then the Bayesian method is used to estimate the output layer weights. In this test, the number of neurons is  $N = 180$  and the number of training points is  $N_f = 400$  and  $N_b = 28$ . The noise scale on the boundary is  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . Fig.10 illustrates the results of PIELM and BPIELM, where the data noise scale on the boundary is  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . BPIELM makes more accurate predictions for the solution than PIELM and gives the uncertainty quantification for the solution. More concretely, MAEs of PIELM and BPIELM are 0.0787 and 0.0186, respectively. The absolute errors are concentrated over the area near  $t = 0$  but also spread all over the spatial-temporal domain. Similarly, for BPIELM, the area near  $t = 0$  has a relatively large standard deviation. To compare two different noise scales, i.e.,  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$  and  $\epsilon_b \sim \mathcal{N}(0, 0.5^2)$ , Fig.11 provides the results between PIELM and BPIELM at three temporal snapshots  $t = 0.25, 1, 2$ . As we can see in the figure, the errors of BPIELM are mostly bounded by two standard deviations, and the errors between predictions and the exact solution increases with the increasing noise scale. In particular, when it comes to the large noise scale, the performance of PIELM is much worse than BPIELM.

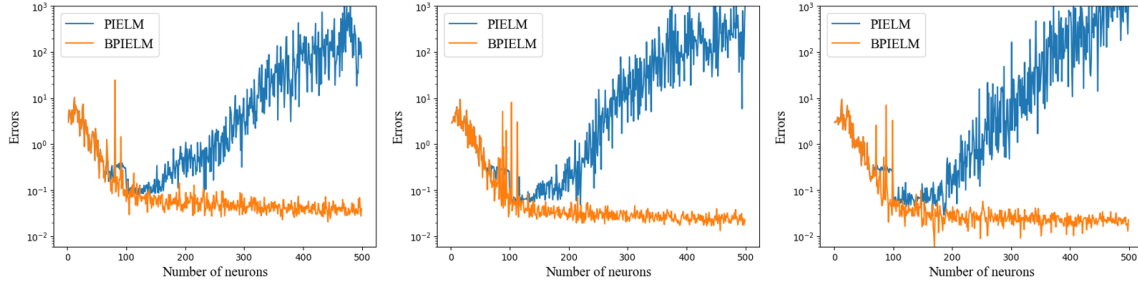
To study the effect of the different number of sensors on the boundary for PIELM and BPIELM,



**Fig. 11.** Diffusion equation: The columns (from left to right) represent the comparisons between BPIELM and PIELM at  $t = -0.25$ ,  $t = 0.1$  and  $t = 2$ , where  $N = 180$ ,  $N_f = 400$  and  $N_b = 30$  are used. The rows (from top to down) represent two noise scales on the boundary,  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$  and  $\epsilon_b \sim \mathcal{N}(0, 0.5^2)$ .



**Fig. 12.** Diffusion equation: The first column represents the maximum error and MAE under different number of measuring points on the boundary, where  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ ,  $n = 180$  and  $N_f = 400$  are used. The second and third columns represent the predictions at  $N_b = 16$  and its absolute error. The rows (from top to down) represent the results of PIELM and BPIELM. Black circles represent the positions of sensors.



**Fig. 13.** Diffusion equation: The columns (from left to right) represent the MAE comparisons between BPIELM and PIELM at  $N_{bc} = 42$ ,  $N_{bc} = 62$  and  $N_{bc} = 92$ , where  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$  and  $N_f = 400$  are used.

**Table 4**

Diffusion equation: The comparisons of PINN, PIELM and BPIELM under three noise scales. The number of training points is  $N_f = 400$ ,  $N_b = 28$  and the number of neurons is  $N = 180$ . PINN has four hidden layers with 100 neurons in each layer and the epoch is 4,000.

Noise scale	0.01			0.05			0.1		
	Max-AE	MAE	Time/s	Max-AE	MAE	Time/s	Max-AE	MAE	Time/s
PINN	0.082	0.019	181	0.138	0.032	182	0.182	0.033	182
PIELM	0.163	0.025	0.84	0.179	0.040	0.93	0.226	0.079	0.91
BPIELM	0.033	0.007	1.67	0.115	0.021	1.76	0.118	0.019	1.68

we compare the results in Fig.12. In this test, the number of neurons is  $N = 180$ , the number of training points is  $N_f = 400$ , and the noise scale is  $\epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . As the first column shows, BPIELM outperforms PIELM in terms of Max-AEs and MAEs under the same number of sensors. Max-AEs and MAEs of PIELM and BPIELM are both worse when the number of sensors is extremely small, while MAEs of BPIELM tend to be less than  $10^{-1}$  when the number of sensors is more than 13. In particular, the results of PIELM and BPIELM with 16 sensors are shown in the last two columns, where BPIELM provides much more predictions than PIELM.

Fig.13 illustrates the effect of the different neuron number for a fixed number of training points in PIELM and BPIELM on the predictive accuracy. As the figure shows, the MAE curve of PIELM is growing up when  $N > 200$ , while the MAE curve of BPIELM converges to a small value and fluctuates in a small range when the neuron number is large enough. It indicates that the solution accuracy of PIELM is more sensitive to the number of neurons than that of BPIELM.

In Table 4, we provide further comparisons of PINN, PIELM and BPIELM in terms of the accuracy and the computational cost under three noise scales. The computational cost of BPIELM and PIELM is comparable, while BPIELM are more accurate than PIELM. Besides, compared to PINN, the training time of BPIELM and PIELM has been reduced by two orders of magnitude.

#### 4.4. Inverse problem

In this subsection, we consider quickly identifying unknown problem parameters in the PDEs by BPIELM. In current research works, PIELM has not been extended to inverse PDE problems, but PIELM can also consider problem parameters as the output layer weights like BPIELM and then uses Moore–Penrose generalized inverse to solve the output layer weights for inverse PDE problems.

##### 4.4.1. Poisson equation

As the first inverse problem test, we consider the following linear Poisson equation,

$$\begin{aligned} u_{xx} &= -\lambda_1 \cdot \sin(0.7x) - \lambda_2 \cdot \cos(1.5x), \quad x \in \Omega, \\ u(x_1) &= b_1, \quad u(x_2) = b_2, \end{aligned} \tag{31}$$

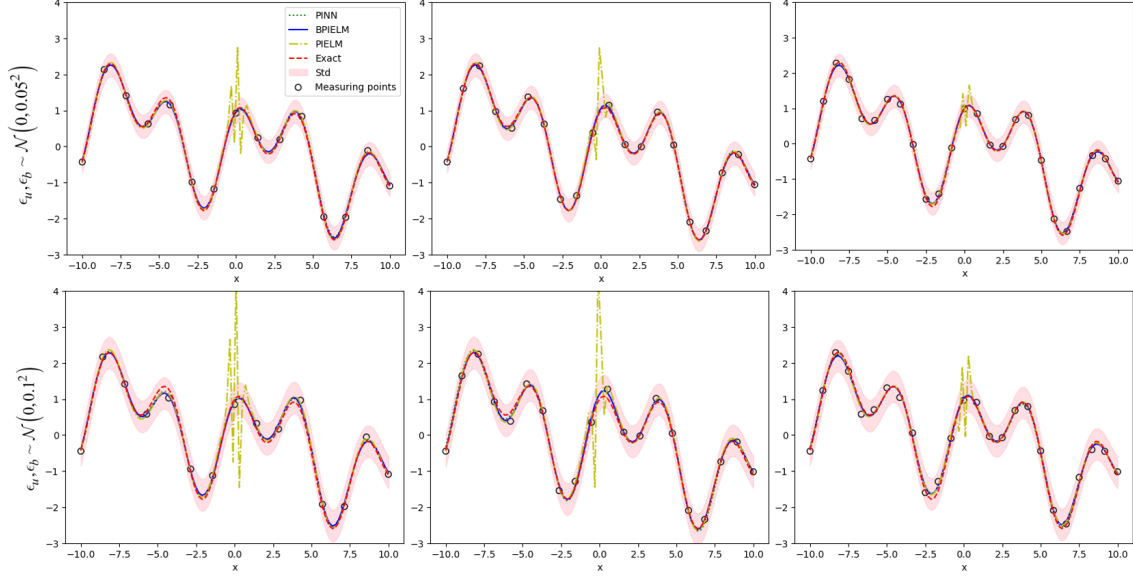
where problem parameters ( $\lambda_1$  and  $\lambda_2$ ) are unknown,  $x_1$  and  $x_2$  are boundary points, and  $\Omega = \{x \mid x \in [-10, 10]\}$ . The exact solution is considered in this problem,

$$u = \sin(0.7x) + \cos(1.5x) - 0.1x. \tag{32}$$

Assuming that the exact solution is unknown, we have  $N_u$  sensors for  $u$ , which are equidistantly distributed in  $\Omega$ . Besides, two sensors are placed at  $x = -10$  and  $x = 10$  to provide the boundary conditions. The Gaussian noises in the measurements are considered, i.e.,  $\epsilon_u$  and  $\epsilon_b$ .

Let us compare BPIELM with PIELM and PINN for solving the Poisson equation. Fig.14 compares the solution using BPIELM, PIELM and PINN under two data noise scales. In the setting of PINN, the NN has three hidden layers with 50 neurons in each layer and the tanh activation function is used. The training epoch with Adam is 4,000. For PIELM and BPIELM, the neuron number is  $N = 100$  and the number of training points is  $N_f = 100$  and  $N_b = 2$ . As shown in the figure, BPIELM gives the uncertainty quantity and provides accurate predictions. The errors of BPIELM are mostly bounded by two standard deviations, which increase as the noise scale increases. Besides, the predictions of PINN are close to BPIELM, which are better than PIELM. Under noise scenarios, PIELM using Moore–Penrose generalized inverse with limited sensor measurements is prone to overfit noisy data, causing worse predictions. The columns from left to right in the figure represent the predictions with  $N_u = 13, 18, 23$ . The predictions with more sensor measurements are better than those with fewer sensor measurements. Besides, the larger





**Fig. 14.** Poisson equation - inverse problem: The comparisons of PINN, PIELM and BPIELM with  $N_u = 13, 18, 23$  under two noise scales. The number of neurons is  $N = 100$  and the number of training points is  $N_f = 100$  and  $N_b = 2$ . Black circles represent the positions of sensors.

**Table 5**

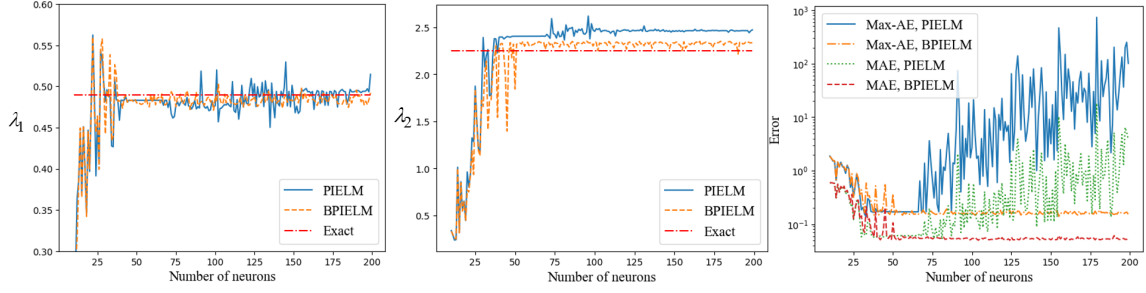
1D Poisson equation - inverse problem: The comparisons of PINN, PIELM and BPIELM under two noise scales. The number of neurons is  $N = 100$  and the number of training points is  $N_f = 100$ ,  $N_u = 18$ ,  $N_b = 2$ . The exact solutions for  $\lambda_1$  and  $\lambda_2$  are 0.49 and 2.25, respectively.

Method	$\epsilon_u, \epsilon_b \in 0.05$					$\epsilon_u, \epsilon_b \in 0.1$				
	$\lambda_1$	$\lambda_2$	Max-AE	MAE	Time/s	$\lambda_1$	$\lambda_2$	Max-AE	MAE	Time/s
PINN	0.491	2.307	0.073	0.027	225.30	0.488	2.370	0.160	0.068	240.12
PIELM	0.484	2.390	1.808	0.081	0.71	0.477	2.280	3.619	0.168	0.77
BPIELM	0.490	2.300	0.076	0.027	2.51	0.480	2.300	0.155	0.054	2.53

noise in the data also causes worse predictions.

Table 5 provides further comparison with PIELM, BPIELM and PINN for both cases with noise scale, i.e.,  $\epsilon_u, \epsilon_b \sim \mathcal{N}(0, 0.05^2)$  and  $\epsilon_u, \epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . The problem setting here is the same as the Fig.14. The data in the table shows that BPIELM is much more accurate than PIELM and is dramatically faster to train PINN (by two orders of magnitude). BPIELM and PINN give close predictions. Besides, we observe that larger noise leads to worse predictions of the problem parameters and the solution. In a word, the results show the effectiveness of BPIELM in identifying the unknown problem parameters and solving the solution from the scatter noisy data.

Fig.15 demonstrates the effect of the different neuron number for a fixed number of training



**Fig. 15.** 1D Poisson equation - inverse problem: The comparisons of PIELM and BPIELM under the different neuron number. The first two columns represent the predictions for the system parameters,  $\lambda_1$  and  $\lambda_2$ , where the exact solutions for  $\lambda_1$  and  $\lambda_2$  are 0.49 and 2.25. The last column represents the comparisons of Max-AEs and MAEs of PIELM and BPIELM. The number of sensors is  $N_u = 18$ , the number of neurons is  $N = 100$  and the number of training points is  $N_f = 100$  and  $N_b = 2$ .

points in PIELM and BPIELM on identifying problem parameters and predicting the solution. The data in the first two columns show that BPIELM are more stable for the neuron number, while PIELM is more sensitive to the neuron number. Note that when the neuron number is large enough, the predictive problem parameters of BPIELM converge to exact values and fluctuate in a small range. As the last column shows, Max-AEs as well as MAEs of PIELM and BPIELM are close under the small neuron number and then PIELM achieves the best Max-AEs and MAEs within a certain neuron number, but it is difficult for PIELM to find appropriate neuron number in practice except for the experiments. However, BPIELM alleviates overfitting noisy data and a large enough number of neurons is chosen for BPIELM, which is more appropriate for noise scenarios.

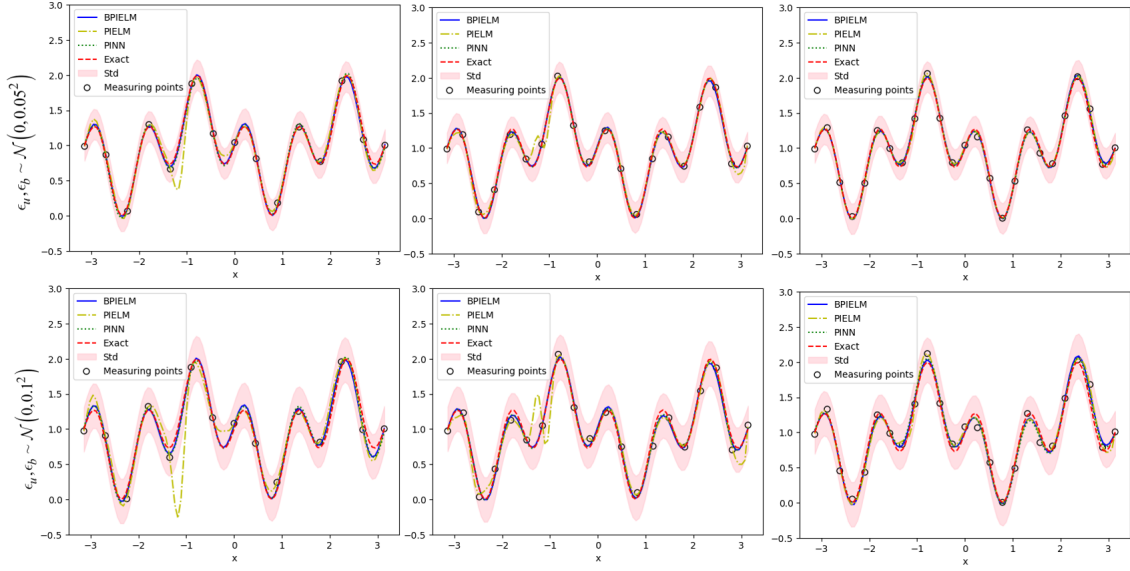
#### 4.4.2. Helmholtz equation

In the next test, we consider the following 1D linear Helmholtz equation

$$\begin{aligned} u_{xx} + k^2 u &= -\lambda_1 f_1(x) + \lambda_2 \cdot f_2(x) - \lambda_3, x \in \Omega, \\ u(x_1) &= b_1, \quad u(x_2) = b_2, \end{aligned} \tag{33}$$

where the problem parameters ( $\lambda_1$ ,  $\lambda_2$  and  $\lambda_3$ ) are unknown, the wave number is  $k^2 = 10$ ,  $x_1$  as well as  $x_2$  are boundary points,  $f_1(x) = \sin(2x) \cos(4x)$ ,  $f_2(x) = \cos(2x) \sin(4x)$ , and  $\Omega = \{x \mid x \in [-2\pi, 2\pi]\}$ . The exact solution is considered in this problem,

$$u = \sin(2x) \cos(4x) + 1. \tag{34}$$

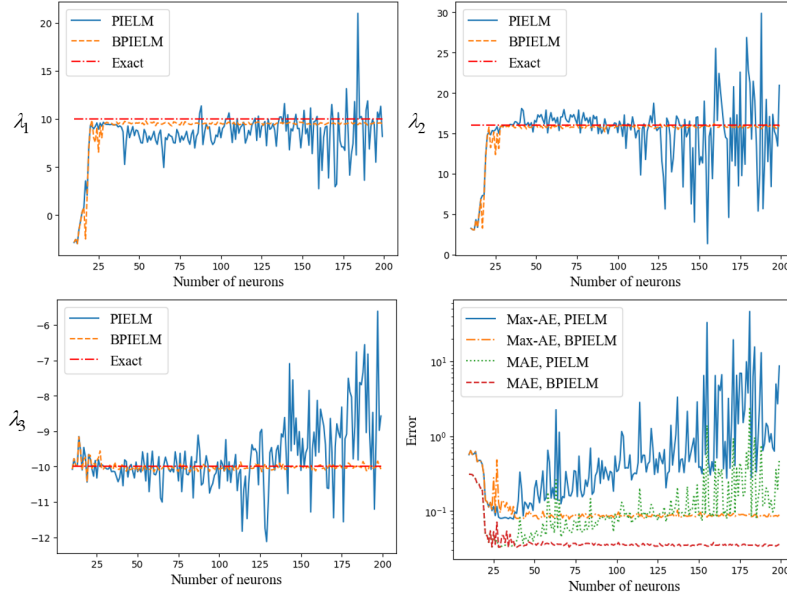


**Fig. 16.** Helmholtz equation - inverse problem: The comparisons of PINN, PIELM and BPIELM with  $N_u = 13, 18, 23$  under two noise scales. The number of training points is  $N_f = 100$  and  $N_b = 2$ . Black circles represent the positions of sensors.

Assuming that the exact solution is unknown, we have  $N_u$  sensors for  $u$ , which are equidistantly distributed in  $\Omega$ . Besides, two sensors are placed at  $x_1 = -2\pi$  and  $x_2 = 2\pi$  to provide the boundary conditions for  $u$ . The Gaussian noises in the measurements are considered, i.e.,  $\epsilon_u$  and  $\epsilon_b$ .

We next compare BPIELM with PIELM and PINN for solving the Helmholtz equation, where more unknown problem parameters are identified. Fig.16 compares the solution using BPIELM, PIELM and PINN under two data noise scale, i.e.,  $\epsilon_u, \epsilon_b \sim \mathcal{N}(0, 0.05^2)$  and  $\epsilon_u, \epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . In the setting of PINN, we use four hidden layers with 100 neurons in each layer and use 4,000 epochs to train the NN with the tanh activation function. For PIELM and BPIELM, the number of neurons is also set to be  $N = 100$ . As the figure shows, BPIELM over PIELM and PINN has a two-fold benefit: (1) BPIELM gives the uncertainty quantity for the solution and errors are mostly bounded by two standard deviations. (2) BPIELM provides more accurate predictions for the solution than PIELM and BPIELM reduces the computational cost to achieve close or better predictions than PINN (the computational cost is shown in Table 6).

Table 6 provides further comparison with PIELM, BPIELM and PINN for both cases with noise scale, i.e.,  $\epsilon_u, \epsilon_b \sim \mathcal{N}(0, 0.05^2)$  and  $\epsilon_u, \epsilon_b \sim \mathcal{N}(0, 0.1^2)$ . The problem setting here is the same as the Fig.16. The results show the effectiveness of BPIELM in identifying the unknown problem



**Fig. 17.** 1D Helmholtz equation - inverse problem: The comparisons of PIELM and BPIELM under the different neuron number. The first row represents the predictions for the system parameters,  $\lambda_1$  and  $\lambda_2$ . The second row (from left to right) represents the prediction for the system parameter  $\lambda_3$  and comparisons of Max-AEs and MAEs. The number of neurons is  $N = 100$  and the number of training points is  $N_f = 100$  and  $N_b = 2$ . The exact solutions for  $\lambda_1$ ,  $\lambda_2$  and  $\lambda_3$  are 10, 16, -10, respectively.

**Table 6**

1D Helmholtz equation - inverse problem: The comparisons of PINN, PIELM and BPIELM under two noise scales. The number of neurons is  $N = 100$  and the number of training points is  $N_f = 100$ ,  $N_u = 18$ ,  $N_b = 2$ . The exact solutions for  $\lambda_1$ ,  $\lambda_2$  and  $\lambda_3$  are 10, 16, -10, respectively.

Method	$\epsilon_u, \epsilon_b \in 0.05$						$\epsilon_u, \epsilon_b \in 0.1$					
	$\lambda_1$	$\lambda_2$	$\lambda_3$	Max-AE	MAE	Time/s	$\lambda_1$	$\lambda_2$	$\lambda_3$	Max-AE	MAE	Time/s
PINN	9.84	15.84	-9.96	0.034	0.015	399.38	9.71	15.71	-9.93	0.071	0.031	398.13
PIELM	9.71	14.8	-10	0.337	0.047	0.69	9.41	13.5	-10	0.683	0.096	0.81
BPIELM	9.80	16.0	-10	0.051	0.018	2.59	9.60	15.9	-10	0.091	0.035	2.51

parameters and solving the solution from the scatter noisy data. BPIELM is not only much more accurate than PIELM, but also considerably cheaper than PINN in terms of the computational cost. The training time with the BPIELM is on the order of two seconds. In contrast, it takes 400 seconds to train PINN to obtain close predictions.

Fig.17 illustrates the effect of different neuron number for a fixed number of training points in PIELM and BPIELM on identifying problem parameters. The results show that PIELM is dramatically more sensitive to the neuron number than BPIELM. For BPIELM, when the neuron number is large enough, predictive problem parameters converge to exact values and fluctuate

in a small range, which indicates the BPIELM avoids overfitting noisy data. For Max-AE and MAE curves, under the small neuron number, MAEs between PIELM and BPIELM are close and worse. It means that a shallow NN architecture both for PIELM and BPIELM can not learn enough physics information to depict the whole solution. As the neuron number increases, MAEs of PIELM increase after reaching the minimum value in a certain range, while the Max-AE and the MAE curves of BPIELM converge to the values and fluctuate in a small range. In summary, considering the accuracy as well as the sensibility to the neuron number, BPIELM is a better approach than PIELM for noise scenarios.

## 5. Conclusion

In this work, we propose the BPIELM framework to quantify the uncertainty arising from noisy data for forward and inverse PDE problems. The physics laws (PDE, boundary conditions, and measurements) are first incorporated into the NN as the cost function. Then a prior probability distribution is introduced in the output layer and the Bayesian method is used to estimate the posterior. To support our claim, we examine BPIELM for forward and inverse PDE problems, including Poisson, advection, diffusion and Helmholtz equations. In those tests under noise scenarios, the results show that BPIELM has a three-fold benefit: (1) BPIELM provides the uncertainty quantification for the solution and more accurate predictions for solutions as well as problem parameters than PIELM. (2) PIELM is prone to be drastically sensitive to the number of hidden neurons, but BPIELM alleviates the sensitivity characteristic and improves the robustness of the NN. (3) BPIELM is considerably cheaper than PINN in terms of the computational cost.

Although BPIELM has exhibited some superior potential for designing fast and accurate scientific machine learning techniques, some things need to be resolved. The high efficiency of PIELM benefits from the linear nature of the final problem, therefore BPIELM, which inherits some characteristics of PIELM, is also limited to studying linear problems. How to further extend and improve this method is an interesting problem, and will be explored in future works.

## CRedit authorship contribution statement

**Xu Liu:** Software, Methodology, Formal analysis, Writing - original draft. **Wen Yao:** Supervision, Funding acquisition, Project administration. **Wei Peng:** Conceptualization, Supervision. **Weien Zhou:** Software, Supervision, Data curation, Visualization, Funding acquisition.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This work was supported by National Natural Science Foundation of China under Grant No.11725211, 52005505, and 62001502.

## References

- [1] A. F. Psaros, X. Meng, Z. Zou, L. Guo, G. E. Karniadakis, Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons, arXiv preprint arXiv:2201.07766.
- [2] M. Alber, A. Buganza Tepole, W. R. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. Karniadakis, W. W. Lytton, P. Perdikaris, L. Petzold, et al., Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences, *NPJ digital medicine* 2 (1) (2019) 1–11.
- [3] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, *Nature Reviews Physics* 3 (6) (2021) 422–440.
- [4] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, arXiv preprint arXiv:1711.10561.
- [5] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational physics* 378 (2019) 686–707.
- [6] S. Basir, I. Senocak, Physics and equality constrained artificial neural networks: Application to forward and inverse problems with multi-fidelity data fusion, arXiv preprint arXiv:2109.14860.
- [7] Z. He, F. Ni, W. Wang, J. Zhang, A physics-informed deep learning method for solving direct and inverse heat conduction problems of materials, *Materials Today Communications* 28 (2021) 102719.

- [8] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks, *Computer Methods in Applied Mechanics and Engineering* 358 (2020) 112623.
- [9] X. Jin, S. Cai, H. Li, G. E. Karniadakis, Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations, *Journal of Computational Physics* 426 (2021) 109951.
- [10] H. D. Kabir, A. Khosravi, M. A. Hosen, S. Nahavandi, Neural network-based uncertainty quantification: A survey of methodologies and applications, *IEEE access* 6 (2018) 36218–36234.
- [11] X. Meng, H. Babae, G. E. Karniadakis, Multi-fidelity bayesian neural networks: Algorithms and applications, *Journal of Computational Physics* 438 (2021) 110361.
- [12] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, *Journal of Computational Physics* 394 (2019) 56–81.
- [13] Y. Yang, P. Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, *Journal of Computational Physics* 394 (2019) 136–152.
- [14] L. Yang, D. Zhang, G. E. Karniadakis, Physics-informed generative adversarial networks for stochastic differential equations, *SIAM Journal on Scientific Computing* 42 (1) (2020) A292–A317.
- [15] A. Daw, M. Maruf, A. Karpatne, Pid-gan: A gan framework based on a physics-informed discriminator for uncertainty quantification with physics, in: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 237–247.
- [16] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, M. Bennamoun, Hands-on bayesian neural networks—a tutorial for deep learning users, *arXiv preprint arXiv:2007.06823*.
- [17] E. Goan, C. Fookes, Bayesian neural networks: An introduction and survey, in: *Case Studies in Applied Bayesian Data Science*, Springer, 2020, pp. 45–87.

- [18] A. G. Wilson, P. Izmailov, Bayesian deep learning and a probabilistic perspective of generalization, *Advances in neural information processing systems* 33 (2020) 4697–4708.
- [19] L. Yang, X. Meng, G. E. Karniadakis, B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data, *Journal of Computational Physics* 425 (2021) 109913.
- [20] D. Zhang, L. Lu, L. Guo, G. E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, *Journal of Computational Physics* 397 (2019) 108850.
- [21] J. N. Reddy, *Introduction to the finite element method*, McGraw-Hill Education, 2019.
- [22] R. J. LeVeque, *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, SIAM, 2007.
- [23] T. Bandai, T. A. Ghezzehei, Forward and inverse modeling of water flow in unsaturated soils with discontinuous hydraulic conductivities using physics-informed neural networks with domain decomposition, *Hydrology and Earth System Sciences Discussions* (2022) 1–42.
- [24] Y. Yang, M. Hou, J. Luo, A novel improved extreme learning machine algorithm in solving ordinary differential equations by legendre neural network methods, *Advances in Difference Equations* 2018 (1) (2018) 1–24.
- [25] H. Sun, M. Hou, Y. Yang, T. Zhang, F. Weng, F. Han, Solving partial differential equation based on bernstein neural network and extreme learning machine algorithm, *Neural Processing Letters* 50 (2) (2019) 1153–1172.
- [26] V. Dwivedi, B. Srinivasan, Physics informed extreme learning machine (pielm)—a rapid method for the numerical solution of partial differential equations, *Neurocomputing* 391 (2020) 96–118.
- [27] V. Dwivedi, N. Parashar, B. Srinivasan, Distributed learning machines for solving forward and inverse problems in partial differential equations, *Neurocomputing* 420 (2021) 299–316.
- [28] F. Calabrò, G. Fabiani, C. Siettos, Extreme learning machine collocation for the numerical solution of elliptic pdes with sharp gradients, *Computer Methods in Applied Mechanics and Engineering* 387 (2021) 114188.



- [29] S. Dong, Z. Li, Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations, *Computer Methods in Applied Mechanics and Engineering* 387 (2021) 114129.
- [30] E. Schiassi, R. Furfaro, C. Leake, M. De Florio, H. Johnston, D. Mortari, Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations, *Neurocomputing* 457 (2021) 334–356.
- [31] X. Luo, A. Kareem, Bayesian deep learning with hierarchical prior: Predictions from limited and noisy data, *Structural Safety* 84 (2020) 101918.
- [32] E. Soria-Olivas, J. Gomez-Sanchis, J. D. Martin, J. Vila-Frances, M. Martinez, J. R. Magdalena, A. J. Serrano, Belm: Bayesian extreme learning machine, *IEEE transactions on neural networks* 22 (3) (2011) 505–509.
- [33] Z. Jin, G. Zhou, D. Gao, Y. Zhang, Eeg classification using sparse bayesian extreme learning machine for brain–computer interface, *Neural Computing and Applications* 32 (11) (2020) 6601–6609.
- [34] T. Chen, E. Martin, Bayesian linear regression and variable selection for spectroscopic calibration, *Analytica chimica acta* 631 (1) (2009) 13–21.
- [35] D. J. MacKay, Probable networks and plausible predictions—a review of practical bayesian methods for supervised neural networks, *Network: computation in neural systems* 6 (3) (1995) 469.
- [36] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980*.