2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Distributed Bayesian Parameter Inference for Physics-Informed Neural Networks

He Bai Kinjal Bhar Jemin George Carl Busart

Abstract— **We consider a distributed Bayesian parameter inference problem where a networked set of agents collaboratively infer the posterior distribution of unknown parameters in a partial differential equation (PDE) based on their noisy measurements of the PDE solution. Given the unknown parameters residing in a known compact set, we assume that a physics-informed neural network (PINN) has already been trained as the prior model, which is valid for all possible parameters within the set. PINNs incorporate PDEs as training constraints for better generalization even with less training samples. We introduce a distributed Langevin Markov Chain Monte-Carlo algorithm that employs the trained PINN model and the agents' noisy measurements to approximate the posterior distribution of the unknown parameters. We establish convergence properties of the algorithm and demonstrate the effectiveness of the proposed approach through numerical simulations.**

Index Terms— **Distributed Bayesian learning, Physics-informed Neural Network, Multi-agent systems**

I. Introduction

Modeling and analysis of complex physical, biological and chemical systems have been hindered due to the prohibitive cost of data acquisition. In the small data regime, majority of state-of-the-art machine learning techniques are not robust, fail to generalize, and do not provide performance guarantees when dealing with such complex systems. However, partial knowledge of physical laws or empirically validated rules governing the dynamics of such complex systems may be available. Recent studies show that this prior knowledge can magnify the information content of limited data when it's incorporated into a learning algorithm. This prior information constrains the admissible solution space to a manageable size which enables the algorithm to quickly steer toward the right solution and generalize well when only a few training examples are available.

Gaussian processes regression (GPR) [1] and physics-informed neural networks (PINNs) [2] have been proposed as powerful alternatives to purely data-driven methods. These approaches model complex systems by encoding the underlying physical laws in the form of differential equations and give relatively accurate predictions for the unknown parameters with limited data. Even though GPRs were initially proposed for linear operators [3], extensions to nonlinear problems were proposed in subsequent studies [4], [5] in the context of both inference and systems identification. However, general applications of GPR for nonlinear partial differential equations (PDEs) often require local linearization and restricting assumptions on the prior which renders the approach brittle [6].

PINNs embed a PDE into the loss of the neural network using automatic differentiation techniques to approximate the solution of the PDE via deep learning [2], [7]–[9]. These algorithms can be used to approximate solutions to both fractional PDEs [10] and stochastic PDEs [11], [12]. Moreover, PINNs can be used to solve *inverse* problems as easily as *forward* problems [13], [14]. In modeling problems with long-time PDE integration, the time-space domain can become prohibitively large. Under such circumstances, a parallel PINN (PPINN) can be used to divide the problem into many independent short-time problems [15]. By utilizing parallel implementation, the PPINN framework further enhances the computational efficiency and speed of the training process.

In Computational Fluid Dynamics and Computational Structural Dynamics, PINNs are used as a function approximator for the solution of the partial differential equations governing the physics of the problem. For example, PINNs are used to solve complex fluid–structural interaction modeling problems such as the aeroelastic blade vibration in turbomachinery [16]. PINNs have also been used to predict a spatiotemporal wind field [17] based on LiDAR measurements.

PINNs are not inherently equipped with uncertainty quantification and their application to noisy data has been largely limited. However, physics-informed generative adversarial networks can be used to quantify parametric uncertainty [11]. Additionally, polynomial chaos expansions can be used to quantify model uncertainty [18]. Bayesian physics-informed neural networks (B-PINN) can be used to solve both forward and inverse PDE problems with noisy data [19]. B-PINNs consist of two parts: a Bayesian neural network (BNN) [20] which uses a PDE prior with unknown terms; and an algorithm for estimating the posterior distributions of the parameters. Specifically, [19] employs the Hamiltonian Monte Carlo [21] and the variational inference [22] to estimate the posterior distributions.

In this paper, we consider an inverse problem where a distributed group of agents (e.g., a sensor network) independently collect and use noisy measurements to identify unknown parameters of a PDE. The existing inverse problem formulations for PINNs, such as [8], [13], [14], [19], estimate the neural network weights, biases, and the unknown

H. Bai and K. Bhar are with Oklahoma State University, Stillwater, OK 74078, USA. he.bai@okstate.edu, kbhar@okstate.edu

J. George and C. Busart are with the U.S. Army Research Laboratory, Adelphi, MD 20783, USA. jemin.george.civ@mail.mil, carl.e.busart.civ@mail.mil

The work of the first two authors were partly supported by the National Science Foundation (NSF) under Grant No. 1925147. The computing for this project was performed at the High Performance Computing Center at Oklahoma State University supported in part through the National Science Foundation grant OAC-1531128.

parameters simultaneously based on the measurements. Such approaches result in a high-dimensional vector which require a large number of measurements. We explore the following alternative strategy to reduce the data requirement.

We start with training a PINN offline for a range of the unknown PDE parameters. We will use the trained PINN as a prior model when estimating the unknown parameters. We then formulate the problem in a distributed Bayesian setting and propose a distributed Langevin Markov Chain Monte Carlo (MCMC) algorithm. This algorithm employs the trained PINN, the noisy measurements collected by the agents, and the samples shared between the agents to approximate the posterior distribution of the unknown parameters. We establish convergence properties of the proposed distributed Bayesian learning algorithm. We expect that our proposed approach requires less online measurements since the PINN weights and biases are fixed during the online estimation of the unknown parameters. Our simulation results demonstrate that the proposed approach is effective in the inference of the unknown parameters.

The rest of this paper is organized as follows. A brief review of PINN is provided in Section II. We formulate the centralized and distributed inference problems in Section III. The proposed distributed Langevin algorithm is presented in Section IV. In Section V, we provide two numerical simulation examples. Conclusions and future work are discussed in Section VI.

Notation: Let $\mathbb{R}^{n\times m}$ denote the set of $n\times m$ real matrices. For a vector $\boldsymbol{\phi}$, ϕ_i is the i^{th} entry of $\boldsymbol{\phi}$. An $n \times n$ identity matrix is denoted as I_n and $\mathbf{1}_n$ denotes an n-dimensional vector of all ones. The p-norm of a vector $\mathbf{x}$ is denoted as $\|\mathbf{x}\|_p$ for $p \in [1, \infty]$. Given matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, $A \otimes B \in \mathbb{R}^{mp\times nq}$ denotes their Kronecker product. For a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ of order n, $\mathcal{V} \triangleq \{v_1, \ldots, v_n\}$ represents the agents or nodes and the communication links between the agents are represented as $\mathcal{E} \triangleq \{e_1, \ldots, e_\ell\} \subseteq \mathcal{V} \times \mathcal{V}$. Let $\mathcal{A} = [a_{i,j}] \in \mathbb{R}^{n\times n}$ be the *adjacency matrix* with entries of $a_{i,j} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and zero otherwise. Define $\Delta = \operatorname{diag}(\mathcal{A}\mathbf{1}_n)$ as the in-degree matrix and $\mathcal{L} = \Delta - \mathcal{A}$ as the graph *Laplacian*. A Gaussian distribution with a mean $\mu \in \mathbb{R}^m$ and a covariance $\Sigma \in \mathbb{R}^{m\times m}_{\geq 0}$ is denoted by $\mathcal{N}(\mu, \Sigma)$.

II. Review of Physics-informed Neural Networks

Consider a generic partial differential equation (PDE) with boundary conditions of the following form

$$\mathcal{P}_y(u; \xi) = F(y; \xi), \quad y \in \mathbb{Y} \tag{1}$$

$$\mathcal{B}_y(u; \xi) = B(y; \xi), \quad y \in \mathbb{B} \tag{2}$$

where y represents the state variables in the PDE, possibly including time t as one of its elements, $\xi \in \mathbb{R}^{d_\xi}$ contains the parameters in the PDE and in the boundary conditions, $u(y)$ is the solution to the PDE, $\mathcal{P}_y(u; \xi)$ is a generic differential operator acting on the entire physical domain $\mathbb{Y}$, $\mathcal{B}_y(u; \xi)$ is the boundary condition operator acting on the boundary domain $\mathbb{B} \subset \mathbb{Y}$, and $F(y; \xi)$ and $B(y; \xi)$ represent the forcing term and the boundary conditions, respectively.

When ξ is known, a physics-informed neural network (PINN) is trained to approximate the PDE solution $u(y)$ [7]–[9]. Denote by θ all the weight matrices and bias vectors in the PINN. Then the PINN output, denoted by $\hat{u}(y; \theta)$, is a nonlinear function of the input y and the neural network parameters θ.

With an explicit form of (1) and (2), the training of the PINN can be achieved by artificially generating data points at $y_i \in \mathbb{Y}$, $i = 1, \cdots, N$ and optimizing θ in the PINN to minimize a physics-informed loss function $\mathcal{O}(\theta)$. One such function can be the minimum square error given by [2]

$$\begin{aligned}\mathcal{O}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathcal{P}_{y_i}(\hat{u}(y_i;\theta);\xi) - F(y_i;\xi)\right\|_2^2 \\ + \frac{1}{M}\sum_{y_i\in\mathbb{B}}\left\|\mathcal{B}_{y_i}(\hat{u}(y_i;\theta);\xi) - B(y_i;\xi)\right\|_2^2\end{aligned} \tag{3}$$

where M is the cardinality of $\mathbb{B}$. To obtain derivatives of $\hat{u}(y; \theta)$, auto-differentiation techniques are used. Stochastic gradient descent type of algorithms can be used to optimize θ. Once the optimization is complete, the PINN $\hat{u}(y; \theta)$ can be used as a surrogate model for (1)–(2). The advantage of the PINN formulation is that synthetic data can be generated for training without collecting real data.

III. Problem formulation

A. Centralized Bayesian inference

In this paper, we investigate the scenario where the parameter ξ is unknown. We propose to first train a PINN $\hat{u}(y, \xi; \theta)$ offline to represent the solution to (1)–(2) for a compact set of $\xi \in \Xi$ and then infer the true parameter ξ_T using the trained PINN model $\hat{u}(y, \xi; \theta)$ and measurements of the PDE solution. Denote by $u(y, \xi)$ the true solution to (1)–(2) for a $\xi \in \Xi$. Then the trained PINN $\hat{u}(y, \xi; \theta)$ can be used as a surrogate model for $u(y, \xi)$ with $y \in \mathbb{Y}$ and $\xi \in \Xi$.

Suppose that N noisy measurements at y_i's, $i = 1, \cdots, N$, of an underlying physical process $u(y_i, \xi_T)$ with $\xi_T \in \Xi$ are collected. The measurements are given by [19]

$$\bar{u}(y_i, \xi_T) = u(y_i, \xi_T) + \epsilon_i^u, \tag{4}$$

where ϵ_i^u is the additive noise with a distribution $p_u(\cdot; \gamma_u)$ in which γ_u contains the known parameters of the noise distribution. It is possible to design inference algorithms to estimate ξ_T given the trained PINN $\hat{u}(y, \xi; \theta)$ and the collected measurements $\bar{u}(y_i, \xi_T)$'s. For example, one can estimate ξ_T as the solution to the optimization problem

$$\begin{aligned}\min_{\xi} \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{u}(y_i,\xi;\theta) - \bar{u}(y_i,\xi_T)\right\|_2^2 \\ \text{subject to } \xi \in \Xi.\end{aligned} \tag{5}$$

However, the formulation in (5) does not capture the uncertainty associated with the final estimate ξ and may result in convergence to a local minimum. Therefore, we adopt a Bayesian approach, where a prior distribution for the unknown parameters ξ is assigned and the posterior distribution of ξ is obtained from the Bayes theorem using the assigned prior and a measurement likelihood function.

The measurement likelihood function depends on the distribution of the noise term ϵ_i^u in (4). Specifically, using $\hat{u}(y_i, \beta; \xi)$ as a surrogate for $u(y_i, \xi)$, we obtain an approximation of the likelihood function as

$$p(\bar{u}(y_i, \xi_T)|\xi) = p_u(\bar{u}(y_i, \xi_T) - u(y_i, \xi); \gamma_u) \tag{6}$$
$$\approx p_u(\bar{u}(y_i, \xi_T) - \hat{u}(y_i, \xi; \theta); \gamma_u). \tag{7}$$

For example, when $\epsilon_i^u \sim \mathcal{N}(0, \sigma^2)$, $p_u(\cdot; \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(\cdot)^2\right)$ and thus $p(\bar{u}(y_i)|\xi) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(\bar{u}(y_i, \xi_T) - u(y_i, \xi))^2\right)$.

Denote the prior for ξ by $p_\xi(\cdot)$. Let $\mathcal{D}$ be the set containing the measurements $\bar{u}(y_i, \xi_T)$, $i = 1, \cdots, N$. The posterior of ξ given $\mathcal{D}$ is given by

$$p(\xi|\mathcal{D}) \propto p_\xi(\xi)p(\mathcal{D}|\xi) = p_\xi(\xi) \prod_{i=1}^{N} p(\bar{u}(y_i, \xi_T)|\xi) \tag{8}$$
$$= p_\xi(\xi) \prod_{i=1}^{N} p_u(\bar{u}(y_i, \xi_T) - u(y_i, \xi); \gamma_u) \tag{9}$$
$$\approx p_\xi(\xi) \prod_{i=1}^{N} p_u(\bar{u}(y_i, \xi_T) - \hat{u}(y_i, \xi; \theta); \gamma_u). \tag{10}$$

It is possible to condition the measurement data $\mathcal{D}$ on both ξ and θ, which leads to simultaneous training of the PINN and estimation of ξ_T. We will examine this option in future work. Note that in this case, the inference problem has a much higher dimension due to the size of θ.

B. Distributed Bayesian inference

Suppose that the N measurements are collected by n agents distributed within $\mathbb{Y}$. Denote by $\mathcal{D}^i$ the measurement data set collected by agent i, $i = 1, \cdots, n$. We assume that $\mathcal{D} = \mathcal{D}^1 \cup \cdots \cup \mathcal{D}^n$ and that $\mathcal{D}^j \cap \mathcal{D}^i = \emptyset$, $\forall i \neq j$. Then the target posterior $p(\xi|\mathcal{D})$ can be written as

$$p(\xi|\mathcal{D}) \propto \prod_{i=1}^{n} p(\mathcal{D}^i|\xi) p_\xi(\xi)^{\frac{1}{n}}. \tag{11}$$

Our objective is for the agents to collaboratively infer the posterior distribution $p(\xi|\mathcal{D})$ by exchanging their samples of ξ. We assume that the agents can communicate information between them and that the communication topology $\mathcal{G}$ is bidirectional and connected.

IV. A Distributed Langevin algorithm

To achieve our objective, we construct a distributed Markov Chain Monte-Carlo (MCMC) algorithm to approximately sample from the posterior. It follows from (11) that

$$\log p(\xi|\mathcal{D}) = \sum_{i=1}^{n} \log p(\mathcal{D}^i|\xi) + \log p_\xi(\xi) + C \tag{12}$$

for some constant C. Following [23]–[26], we further define energy-like functions $U^i(\xi, \mathcal{D}^i) \geq 0$ and $V(\xi) \geq 0$ such that

$$p(\mathcal{D}^i|\xi) \propto \exp(-U^i(\xi, \mathcal{D}^i)) \quad \text{and} \quad p_\xi(\xi) \propto \exp(-V(\xi)). \tag{13}$$

Note that $U^i(\xi, \mathcal{D}^i)$ can be computed based on the collected data $\mathcal{D}^i$ and ξ. Let $Y^i(\xi, \mathcal{D}^i) = U^i(\xi, \mathcal{D}^i) + \frac{1}{n}V(\xi)$. It follows from (13) that there exists a constant C_1 such that

$$-\log p(\mathcal{D}^i|\xi) - \frac{1}{n} \log p_\xi(\xi) = Y^i(\xi, \mathcal{D}^i) + C_1. \tag{14}$$

The Unadjusted Langevin Algorithm (ULA) [23]–[26] has been proposed as a first-order, gradient based Monte Carlo sampling technique to approximate the posterior as

$$\xi(k+1) = \xi(k) - \alpha \sum_{i=1}^{n} \nabla_\xi Y^i(\xi(k), \mathcal{D}^i) + \sqrt{2\alpha} v(k), \tag{15}$$

where $\alpha > 0$ is the step-size and $v(k) \sim \mathcal{N}(0, I_{d_\xi})$. For a sufficiently small α, (15) guarantees that the Kullback–Leibler (KL) divergence of the distribution of $\xi(k)$ with respect to $p(\xi|\mathcal{D})$ converges to a small bias proportional to α. Substituting $Y^i(\xi, \mathcal{D}^i) = U^i(\xi, \mathcal{D}^i) + \frac{1}{n}V(\xi)$ yields

$$\xi(k+1) = \xi(k) - \alpha \sum_{i=1}^{n} \nabla_\xi U^i(\xi(k), \mathcal{D}^i) - \alpha \nabla_\xi V(\xi(k)) + \sqrt{2\alpha} v(k) \tag{16}$$
$$\approx \xi(k) - \alpha \sum_{i=1}^{n} \mathbf{g}^i\left(\xi(k), \mathcal{D}^i\right) - \alpha \nabla_\xi V(\xi(k)) + \sqrt{2\alpha} v(k), \tag{17}$$

where $\mathbf{g}^i\left(\xi(k), \mathcal{D}^i\right)$ is the approximation of $\nabla_\xi U^i(\xi(k), \mathcal{D}^i)$ in which the PINN $\hat{u}(\cdot, \xi; \theta)$ is used in the likelihood computation as in (7). Specifically,

$$\mathbf{g}^i\left(\xi(k), \mathcal{D}^i\right) = -\sum_{j=1}^{|\mathcal{D}^i|} \left[\nabla_\xi \log p_u\Big(\bar{u}(y_j, \xi_T) - \hat{u}(y_j, \xi(k); \theta); \gamma_u\Big)\right]. \tag{18}$$

A. Distributed Langevin algorithm

We choose the prior $p_\xi(\xi) \sim \mathcal{N}(\mu, s^{-1}I_{d_\xi})$ for any $\mu \in \mathbb{R}^{d_\xi}$ and $s > 0$. According to (13), we set

$$V(\xi) = \frac{s}{2}\|\xi - \mu\|^2. \tag{19}$$

Let $\boldsymbol{\xi}^i(k) \in \mathbb{R}^{d_\xi}$ denote the k-th posterior sample of ξ for agent i, $i = 1, \cdots, n$. We propose the following distributed Langevin algorithm

$$\boldsymbol{\xi}^i(k+1) = \boldsymbol{\xi}^i(k) - \beta \sum_{j=1}^{n} a_{i,j}(\boldsymbol{\xi}^i(k) - \boldsymbol{\xi}^j(k)) - \alpha \nabla_\xi V(\boldsymbol{\xi}^i(k)) - \alpha n \mathbf{g}^i\left(\boldsymbol{\xi}^i(k), \mathcal{D}^i\right) + \sqrt{2\alpha} \boldsymbol{v}^i(k), \tag{20}$$

where $a_{i,j}$ denotes the entries of the adjacency matrix corresponding to the communication network $\mathcal{G}(\mathcal{V}, \mathcal{E})$, β is the consensus step-size and $\boldsymbol{v}^i(k) \sim \mathcal{N}(\mathbf{0}, nI_{d_\xi})$.

The proposed algorithm (20) is distributed since each agent employs information only from itself and its neighbors. The agents share a common PINN $\hat{u}(\cdot, \xi; \theta)$ and a common prior $p_\xi(\cdot)$. To collect samples from the posterior distribution $p(\xi|\mathcal{D})$, multiple Markov chains can be run in parallel for each agent and the final sample from each chain can be retained as an approximation of $p(\xi|\mathcal{D})$. Another approach is to collect samples from a single Markov chain after a transient period. The algorithm is generic and can be applied to other distributed inference problems. The psuedo code of the proposed algorithm is given in Algorithm 1.

To analyze the algorithm, we define

$$\epsilon(\boldsymbol{\xi}^i(k), \mathcal{D}^i) = \mathbf{g}^i\left(\boldsymbol{\xi}^i(k), \mathcal{D}^i\right) - \nabla_\xi U^i(\boldsymbol{\xi}^i(k), \mathcal{D}^i), \quad (21)$$

where

$$\nabla_\xi U^i(\boldsymbol{\xi}^i(k), \mathcal{D}^i) = -\sum_{j=1}^{|\mathcal{D}^i|}\Big[\nabla_\xi \log p_u(\bar{u}(y_j, \xi_T) - u(y_j, \boldsymbol{\xi}^i(k)); \gamma_u)\Big]. \quad (22)$$

The $\epsilon(\boldsymbol{\xi}^i(k), \mathcal{D}^i)$ represents the error induced by the PINN approximation. We rewrite (20) using $\nabla Y^i = \frac{1}{n}\nabla V + \nabla U^i$ and (21) as

$$\begin{aligned}\boldsymbol{\xi}^i(k+1) \; &= \; \boldsymbol{\xi}^i(k) - \beta\sum_{j=1}^{n} a_{i,j}(\boldsymbol{\xi}^i(k) - \boldsymbol{\xi}^j(k)) \\ &- \alpha n \nabla_\xi Y^i(\boldsymbol{\xi}^i(k), \mathcal{D}^i) + \alpha n \epsilon(\boldsymbol{\xi}^i(k), \mathcal{D}^i) + \sqrt{2\alpha}\boldsymbol{v}^i(k).\end{aligned} \quad (23)$$

If $\epsilon(\boldsymbol{\xi}^i(k), \mathcal{D}^i) = 0$, (23) is similar to the distributed unadjusted Langevin algorithm with time-varying step-sizes in [27]. Due to the page limit, we omit the convergence analysis of (23) and provide a brief summary of the convergence results. Consensus between the agents is established with the expected second moment of the consensus error bounded by a vanishing term and a constant offset proportional to α. The offset accounts for the added noise $\mathbf{v}_k$, the second moment of the gradient difference, and the PINN's gradient approximation error. Convergence of the averaged sample's distribution to the target distribution $p(\xi|D)$ is proved by showing that its KL divergence is upper bounded by the sum of an exponentially decaying term and a constant offset C_{F_1} proportional to $\alpha^{\frac{1}{2}}$. This upper bound can be improved to $O(\alpha)$ if α is chosen sufficiently small. Assumptions follow from distributed nonconvex optimization, including bounded gradients, Lipschitz smoothness of the likelihood function, and conditions on the step-sizes α and β and on the target distribution $p(\xi|D)$.

Algorithm 1 Decentralized Bayesian Inferencing for PINNs

```
1:  Initialization: $\boldsymbol{\xi}^1(0), \cdots, \boldsymbol{\xi}^n(0)$, $k = 0$
2:  Input: $K > 0$, $\alpha$ and $\beta$
3:  while $k \leq K$ do
4:      for $i = 1$ to $n$ (in parallel) do
5:          Sample $\boldsymbol{v}^i(k) \sim \mathcal{N}(\mathbf{0}, nI_{d_\xi})$
6:          Compute $\mathbf{g}^i\left(\boldsymbol{\xi}^i(k), \mathcal{D}^i\right)$ as in (18)
7:          Compute $\hat{\boldsymbol{\xi}}^i(k) \leftarrow \sum_{j=1}^n a_{i,j}\left(\boldsymbol{\xi}^i(k) - \boldsymbol{\xi}^j(k)\right)$
8:          Update $\boldsymbol{\xi}^i(k+1) \leftarrow \boldsymbol{\xi}^i(k) - \beta\hat{\boldsymbol{\xi}}^i(k) - \alpha\nabla_\xi V(\boldsymbol{\xi}^i(k)) - \alpha n\, \mathbf{g}^i\left(\boldsymbol{\xi}^i(k), \mathcal{D}^i\right) + \sqrt{2\alpha}\boldsymbol{v}^i(k)$
9:      end for
        $k \leftarrow k + 1$
10: end while
```

V. Simulation Examples

We present two examples. In the first example, we illustrate the uncertainty produced by the centralized algorithm (17) and compare it with a gradient descent (GD) algorithm. In the second example, we demonstrate the distributed algorithm (20). All the neural networks used in the examples have 8 layers with 20 neurons in each layer. The activation function is $\tanh$. We use the SCIANN wrapper in [8] for PINN training. The priors for the unknown parameters are the standard normal random distribution truncated to Ξ. Each agent discards final samples that are outside of Ξ.

A. Burger's equation

We consider the 1-D Burger's equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial y} - \frac{\xi}{\pi}\frac{\partial^2 u}{\partial y^2} = 0, \quad t \in [0, 1], \quad y \in [-1, 1], \quad (24)$$

with the boundary conditions

$$u(t, y = \pm 1) = 0, \quad u(t = 0, y) = -\sin \pi y. \quad (25)$$

We train a PINN $\hat{u}(t, y, \xi)$ for $\xi \in \Xi = [0.01, 0.2]$ with uniform grid points of $100 \times 100 \times 10$ in $[0, 1] \times [-1, 1] \times [0.01, 0.2]$. We use a batch size of 2048 with 10000 epochs. Similarly, we train a PINN $u(t, y)$ for the PDE with the true parameter $\xi_T = 0.15$. This is used as the ground truth. Shown in Fig. 1, three sensors are deployed to measure $u(t, y, \xi_T)$ with a zero-mean Gaussian sensor noise of a standard deviation of 0.05.

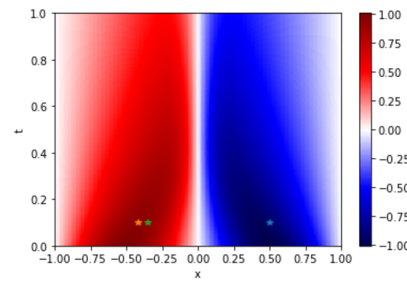

Fig. 1: Three sensors (indicated by stars) are located in the field to measure the true physics.

We run 100 Markov chains in parallel for each agent and collect the final samples. We compare the mean and standard deviation of estimates between the Bayesian inference algorithm (17) and a GD algorithm that maximizes the measurement likelihood. The GD algorithm is initialized with 100 initial estimates that are the same as the initial samples of the 100 Markov chains.

When only one measurement for each sensor is collected at $t = 0.1$ second, as shown in Fig. 2, the GD result converges to a wrong estimate with zero variance, meaning that all 100 initial conditions lead to the same result. However, the Bayesian inference result indicates that there is a large uncertainty in the estimates. When we increase the number of measurements to 10 between $t = 0.1$ and $t = 0.5$, both algorithms produce estimates close to $\xi_T = 0.15$, as shown in Fig. 3. However, the GD results have no uncertain measure.

The uncertainty produced by the Bayesian inference results clearly indicates reduced variance due to the increase of the number of measurements whereas the GD result only produces one estimate without a variance measure.

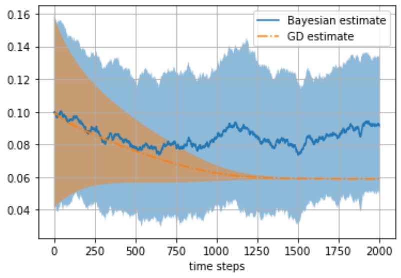

Fig. 2: Comparison of the Bayesian inference performance (variance and mean estimates) with a GD algorithm that maximizes the measurement likelihood using 1 measurement per agent at $t = 0.1$ seconds for each of the three sensors. The spread plot indicates 1 standard deviation.

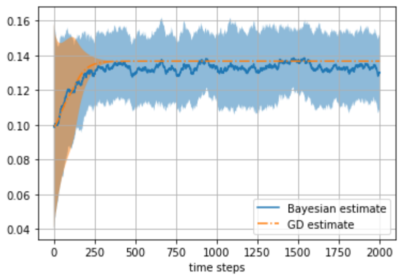

Fig. 3: Comparison of the Bayesian inference performance (standard deviation and mean estimates) with a GD algorithm that maximizes the measurement likelihood using 10 measurements per agent between $t = 0.1$ and $t = 0.5$ seconds for each of the three sensors. The spread plot indicates 1 standard deviation.

B. Transport equation

We consider a two-dimensional transport equation

$$\frac{\partial u}{\partial t} + \xi_v \cdot \nabla u = 0, \tag{26}$$

where $\xi_v = (\xi_x, \xi_y)^\top \in \mathbb{R}^2$ is a fixed unknown velocity. The initial condition is given by $u(x,y,0) = \exp(-\frac{1}{0.02^2}((x-0.05)^2 + (y-0.05)^2))$.

We assume that three sensors are deployed to measure $\xi_c + u(x_i, y_i, t)$, $i = 1, 2, 3$, where ξ_c represents a common unknown bias in the measurements. Fig. 4 shows the location of the sensors. The communication topology is a string graph where agent 2 (located at the top left among the three agents) communicates with agent 1 and agent 3.

The objective is to estimate $\xi = (\xi_c, \xi_x, \xi_y)^\top$ given spatial-temporal measurements of $\xi_c + u(x_i, y_i, t)$. A neural network was trained offline for $(x, y, t, \xi_x, \xi_y) \in [0,1]\times[0,1]\times[0,1]\times[-1,1]\times[-1,1]$. 2.5 million points were uniformly sampled from the domain and used to train a PINN for 3000 epochs.

Each agent collects 5 measurements between $t = 0.1$ and $t = 0.3$ seconds at its location. The measurements are generated from the truth $u(x,y,t) = \xi_c + \exp(-\frac{1}{0.02^2}((x - 0.05 - \xi_x t)^2 + (y - 0.05 - \xi_y t)^2))$ and corrupted by a Gaussian

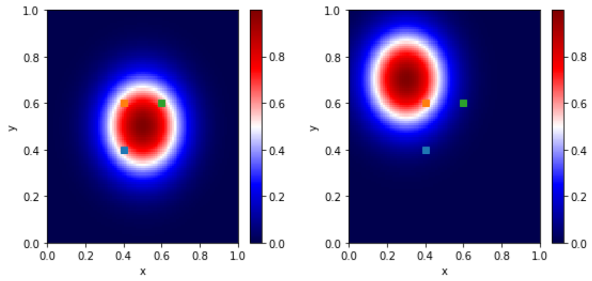

Fig. 4: Three sensors (indicated by squares) are located in the field to measure the field. Left: $t = 0$ second. Right: $t = 0.3$ seconds.

noise $\mathcal{N}(0, 0.01^2)$. We set $\xi_c = 1$, $\xi_x = -0.5$ and $\xi_y = 0.5$ as the truth.

Fig. 5 shows the estimation performance of the three agents. We observe very similar performance across the agents and convergence to the true values with small uncertainty. As a comparison, Fig. 6 shows the centralized estimation performance, where the measurements of the agents are aggregated and processed at a central location. If the agents do not communicate and use only their own measurements, they produce very uncertain estimates, as shown in Fig. 7 for agent 2. In this case, agent 1 and 3 perform worse.

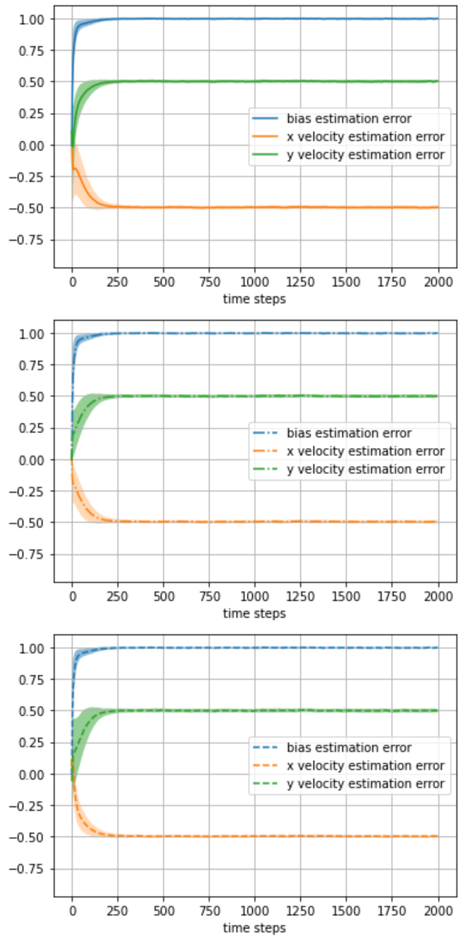

Fig. 5: Distributed estimation performance of ξ_c, ξ_x, and ξ_y for agent 1–3. The spread plots are for 1 standard deviation.

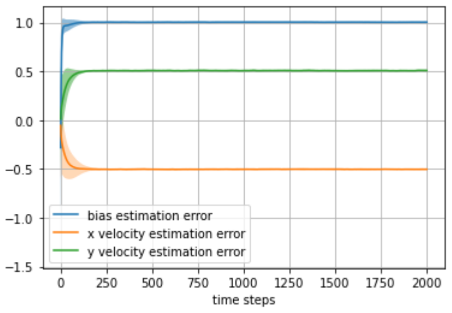

Fig. 6: Centralized inference performance where all the measurements are aggregated.

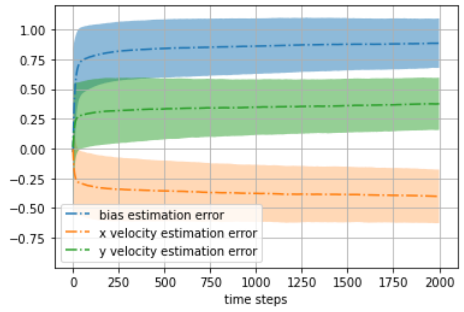

Fig. 7: Without communication, the estimation performance of agent 2 has large uncertainties.

VI. Conclusions and Future Work

In this paper, we investigate a distributed inference problem for a PDE with unknown parameters. We consider that a group of agents measure the PDE solution at distinct locations. We train a physics-informed neural network for a range of the unknown parameters and design a distributed Langevin Markov Chain Monte Carlo algorithm to approximate the posterior distribution of the unknown parameters. We establish convergence properties of the algorithm. Our preliminary simulation results demonstrate the effectiveness of the proposed approach. Our future work involves investigation of different priors and testing on complex PDEs.

References

[1] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Inferring solutions of differential equations using noisy multi-fidelity data," *Journal of Computational Physics*, vol. 335, pp. 736–746, 2017.

[2] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[3] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Machine learning of linear differential equations using gaussian processes," *Journal of Computational Physics*, vol. 348, pp. 683–693, 2017.

[4] ——, "Numerical gaussian processes for time-dependent and nonlinear partial differential equations," *SIAM Journal on Scientific Computing*, vol. 40, no. 1, pp. A172–A198, 2018.

[5] M. Raissi and G. E. Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," *Journal of Computational Physics*, vol. 357, pp. 125–141, 2018.

[6] H. Owhadi, C. Scovel, and T. Sullivan, "Brittleness of Bayesian inference under finite information in a continuous world," *Electronic Journal of Statistics*, vol. 9, no. 1, pp. 1 – 79, 2015.

[7] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, 2020.

[8] E. Haghighat and R. Juanes, "Sciann: A keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 373, p. 113552, 2021.

[9] X. Meng and G. E. Karniadakis, "A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems," *Journal of Computational Physics*, vol. 401, p. 109020, 2020.

[10] G. Pang, L. Lu, and G. E. Karniadakis, "fPINNs: Fractional physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. A2603–A2626, 2019.

[11] L. Yang, D. Zhang, and G. E. Karniadakis, "Physics-informed generative adversarial networks for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 42, no. 1, pp. A292–A317, 2020.

[12] D. Zhang, L. Guo, and G. E. Karniadakis, "Learning in modal space: Solving time-dependent stochastic pdes using physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 42, no. 2, pp. A639–A665, 2020.

[13] Y. Chen, L. Lu, G. E. Karniadakis, and L. D. Negro, "Physics-informed neural networks for inverse problems in nano-optics and metamaterials," *Opt. Express*, vol. 28, no. 8, pp. 11 618–11 633, Apr 2020.

[14] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

[15] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, "PPINN: Parallel physics-informed neural network for time-dependent pdes," *Computer Methods in Applied Mechanics and Engineering*, vol. 370, p. 113250, 2020.

[16] E. Ang and B. F. Ng, "Physics-Informed Neural Networks for the Modelling of Fluid-Structure Interactions," in *APS Division of Fluid Dynamics Meeting Abstracts*, ser. APS Meeting Abstracts, Jan. 2020, p. Q02.004.

[17] J. Zhang and X. Zhao, "Spatiotemporal wind field prediction based on physics-informed deep learning and lidar measurements," *Applied Energy*, vol. 288, p. 116641, 2021.

[18] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis, "Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems," *Journal of Computational Physics*, vol. 397, p. 108850, 2019.

[19] L. Yang, X. Meng, and G. E. Karniadakis, "B-PINNs: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data," *Journal of Computational Physics*, vol. 425, p. 109913, 2021.

[20] R. M. Neal, *Bayesian Learning for Neural Networks*. Springer New York, 1996.

[21] R. M. Neal, "MCMC using Hamiltonian dynamics," *arXiv e-prints, arXiv:1206.1901*, 2012.

[22] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[23] Y.-A. Ma, Y. Chen, C. Jin, N. Flammarion, and M. I. Jordan, "Sampling can be faster than optimization," *Proceedings of the National Academy of Sciences*, vol. 116, no. 42, pp. 20 881–20 885, 2019.

[24] X. Cheng and P. L. Bartlett, "Convergence of Langevin MCMC in KL-divergence," *Proceedings of Machine Learning Research*, no. 83, pp. 186–211, 2018.

[25] S. Vempala and A. Wibisono, "Rapid convergence of the unadjusted Langevin algorithm: Isoperimetry suffices," in *Advances in Neural Information Processing Systems*, 2019, pp. 8094–8106.

[26] M. Raginsky, A. Rakhlin, and M. Telgarsky, "Non-convex learning via stochastic gradient Langevin dynamics: a nonasymptotic analysis," in *Proceedings of the 2017 Conference on Learning Theory*, vol. 65. PMLR, 07–10 Jul 2017, pp. 1674–1703.

[27] A. Parayil, H. Bai, J. George, and P. Gurram, "Decentralized Langevin dynamics for Bayesian learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.