A VARIATIONAL APPROACH TO PHYSICS INFORMED NEURAL NETWORK FOR STOCHASTIC PARTIAL DIFFERENTIAL EQUATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Physics-Informed Neural Networks (PINNs) refer to a deep learning approach to represent the spatial and temporal characteristics of a distributed physical phenomenon, such as thermal fields, using Neural Networks (NNs). The loss function used to train PINNs relies on a term that penalises any violation of the Partial Differential Equation (PDE) that governs the distributed phenomena, in addition to a least-squares-error penalisation of any available observations in the relevant domain. PINNs have been shown to be successful in approximating the solutions to PDEs and have proven effective even in the low data regime. A critical shortcoming is the lack of a systematic treatment of the uncertainty of the approximation. State-of-the-art approaches rely on Bayesian NNs, though these are computationally heavy, both during training and at inference, nor does the associated training procedure derive from first principles. To remedy these limitations, we propose Variational Inference PINNs (VI-PINNs). Our approach derives from first principles: the uncertainty intrinsic to the distributed phenomena is explained by adopting Stochastic PDEs, while relying on standard measurement uncertainty to explain the observational uncertainty. This leads to the formulation of a posterior probability for the distributed phenomenon. Drawing parallels with Bayesian inference in finite spaces and relying on VI techniques to circumvent the otherwise intractable posterior. We derive a training objective that allows us to train two NNs, representing the mean and covariance of the approximation, respectively. The solution may be interpreted as a Bayesian belief about the true distributed phenomenon. Importantly, in the limit, the original PINN framework is recovered. We compare our approach with Bayesian PINNs (B-PINNs). Our results suggest that VI-PINNs are easier to implement, have a lower training time, and yields results that better align with reality, especially when extrapolating outside the measurement range.

Keywords: Neural Network, Variational Inference, Steady State Heat Equation, Physics-Informed Neural Network, Partial Differential Equations, Gaussian Process

1 INTRODUCTION

A Physics-Informed Neural Network (PINN) is a new class of Neural Network (NN) introduced by Raissi et al. (2017). The main idea of a PINN is to use the strengths of a NN, such as approximating (complex) models [Markidis (2021), Bhatnagar et al. (2024), Fowler et al. (2024)], modelling unknown parts of a model [Markidis (2021)], or pattern recognition in data sets [Mah & Chakravarthy (1992)]. That is done in combination with the underlying physics of a system. The underlying physics is introduced trough the use of a Partial Differential Equations (PDEs) or Ordinary Differential Equations (ODEs). This allows for a smaller dataset of measurements, compared to a traditional NN. It can be used to solve forward problems, such as approximating/ emulating solutions to PDEs and ODEs, or inverse problems, such as estimating model parameters from limited data.

PINNs have been shown to yield good results on a broad range of applications [Raissi et al. (2017), Markidis (2021)], but they also have their limitations. PINNs has no uncertainty quantification built into it. This makes PINNs sensitive to the noise in the training data. In Yang et al. (2021),

a Bayesian PINN (B-PINN) is proposed to solve this problem. The uncertainty is quantified using sample-based techniques such as Hamiltonian Monte Carlo (HMC) [Cobb & Jalaian (2021)] or No-U-Turn Sampler (NUTS) [Hoffman & Gelman (2011)]. These are computation-heavy and result in a long training time for each iteration step.

To overcome this problem, we propose a Variational Inference Physics Informed Neural Network (VI-PINN). This topology quantifies the uncertainty in an algebraic way instead of using sampling. This paper is structured as follows: In section 2, the background for PINN and B-PINN is given. In section 3, the VI-PINN is constructed using Variational Inference and Bayesian inference. This is first done for the general finite-dimensional case, followed by an ad hoc generalisation. In section 4, the theoretical framework is implemented in a 1D problem, where it will be compared with the results of a B-PINN. In section 5, conclusions are presented.

2 BACKGROUND

2.1 PHYSICS INFORMED NEURAL NETWORK

The dynamics of a system can be represented using PDEs, which has the general form

$$\mathcal{A}_{\boldsymbol{x},\boldsymbol{t}}[\boldsymbol{u}](\boldsymbol{x},\boldsymbol{t}) = \partial_{\boldsymbol{t}}\boldsymbol{u}(\boldsymbol{x},\boldsymbol{t}) + \mathcal{A}_{\boldsymbol{x}}[\boldsymbol{u}](\boldsymbol{x},\boldsymbol{t}) = 0 \tag{1}$$

$\boldsymbol{x} \in \mathbb{R}^n$ is a spatial vector, $\boldsymbol{t} \in \mathbb{R}$ the time, $\partial_{\boldsymbol{t}}\boldsymbol{u}(\boldsymbol{x},\boldsymbol{t})$ is the partial derivative of $\boldsymbol{u}(\boldsymbol{x},\boldsymbol{t})$ with respect to $\boldsymbol{t}$ and $\mathcal{A}_{\boldsymbol{x}}$ is some spatial linear operator. The objective is to find the explicit formulation $\boldsymbol{u}(\boldsymbol{x},\boldsymbol{t})$ that satisfies equation (1), so that $\boldsymbol{u}(\boldsymbol{x},\boldsymbol{t})$ describes the dynamics in the spatio-temporal domain.

In a PINN, the unknown function $\boldsymbol{u}$ is approximated by the NN, with as output $\boldsymbol{u}_{\boldsymbol{\theta}}$ and NN-parameters $\boldsymbol{\theta}$. The cost function is of the form

$$J_{cost}(\boldsymbol{\theta}) = (1-\lambda)\sum_{(\boldsymbol{x_m},\boldsymbol{t_m},\boldsymbol{y})\in\mathcal{D}_{DATA}} \frac{1}{2}||\boldsymbol{y} - \boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{x_m},t_{\boldsymbol{m}})||^2 + \lambda \sum_{(\boldsymbol{x},\boldsymbol{t})\in\mathcal{D}_{PHYS}} \frac{1}{2}||\mathcal{A}_{\boldsymbol{x},\boldsymbol{t}}[\boldsymbol{u}_{\boldsymbol{\theta}}](\boldsymbol{x},\boldsymbol{t})||^2 \tag{2}$$

$\boldsymbol{y}$ are measurements at $(\boldsymbol{x_m},\boldsymbol{t_m})$, λ is a tuning factor that determines the relative importance of each cost function, $\mathcal{D}_{DATA}$ is the data set, and $\mathcal{D}_{PHYS}$ represent an arbitrary set of space-time coordinates where the PDEs is enforced to the NN. In the literature, there is no clear way to determine the value of λ, nor the set of space-time coordinates.

2.2 BAYESIAN PHYSICS-INFORMED NEURAL NETWORK

A way to quantify the uncertainty of the prediction is by extending the PINN framework to a B-PINN [Yang et al. (2021)]. This can be done by following the Bayesian Neural Network (BNN) approach [Jospin et al. (2022)] and constructing a posterior for the NN-parameters (and optionally the PDEs parameters).

$$p(\boldsymbol{\theta}|\mathcal{D}_{DATA},\mathcal{D}_{PHYS}) \propto p(\mathcal{D}_{DATA}|\boldsymbol{\theta})p(\mathcal{D}_{PHYS}|\boldsymbol{\theta})$$

where

$$p\left(\mathcal{D}_{DATA}|\boldsymbol{\theta}\right) = \prod_{(\boldsymbol{x_m},\boldsymbol{t_m},\boldsymbol{y})\in\mathcal{D}_{DATA}} \mathcal{N}\left(\boldsymbol{y}|\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{x_m},\boldsymbol{t_m}),\sigma_{\boldsymbol{y}}^2\right) \tag{3a}$$

$$p\left(\mathcal{D}_{PHYS}|\boldsymbol{\theta}\right) = \prod_{(\boldsymbol{x},\boldsymbol{t})\in\mathcal{D}_{PHYS}} \mathcal{N}\left(0|\mathcal{A}_{\boldsymbol{x},\boldsymbol{t}}[\boldsymbol{u}_{\boldsymbol{\theta}}](\boldsymbol{x},\boldsymbol{t}),\sigma_{\boldsymbol{u}}^2\right) \tag{3b}$$

Since the posterior is intractable, it can be approximated using Markov Chain Monte Carlo sampling. Possible strategies to establish these samples are HMC or NUTS. In Figure 1, the B-PINN approach is schematically shown.

In the end, one obtains a weighted particle set $\{(\boldsymbol{a}_i,\boldsymbol{\theta}_i)\}$ that approximates $p(\boldsymbol{\theta}|\mathcal{D}_{DATA},\mathcal{D}_{PHYS})$. This particle set can be used to evaluate the output uncertainty of the distribution $\mathbf{u}$, given $(\boldsymbol{x},\boldsymbol{t})$, according to the predicted distribution

$$p(\mathbf{u}|\boldsymbol{x},\boldsymbol{t}) = \int p(\boldsymbol{\theta}|\mathcal{D}_{DATA},\mathcal{D}_{PHYS})\delta\left(\mathbf{u} - \boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{t})\right)d\boldsymbol{\theta} \approx \sum_i \boldsymbol{a}_i\delta\left(\mathbf{u} - \boldsymbol{u}_{\boldsymbol{\theta}_i}(\boldsymbol{x},\boldsymbol{t})\right)$$

and certain statistical measures in particular, e.g.

$$\mu_k(\boldsymbol{x},\boldsymbol{t}) = \int \mathbf{u}^k p(\mathbf{u}|\boldsymbol{x},\boldsymbol{t})d\mathbf{u} \approx \sum_i \boldsymbol{a}_i\boldsymbol{u}_{\boldsymbol{\theta}_i}(\boldsymbol{x},\boldsymbol{t})^k \tag{4}$$

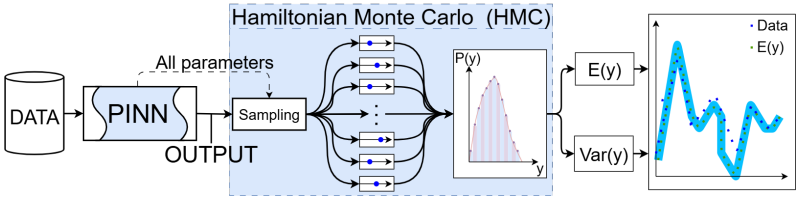


Figure 1: Schematic of a B-PINN using HMC

2.2.1 Implementation of the B-PINN

A B-PINN is capable of finding a good approximation, but it does so inefficiently. It takes N_{MC} samples of the uncertain parameters. This includes all weights and biases from the NN, and optionally the parameters of the PDEs too. All N_{par} uncertain parameters are assumed to have a form of uncertainty [Cobb & Jalaian (2021)]. This results in $N_{MC} \cdot N_{par}$ samples being taken for every iteration step. A proper estimation of the output uncertainty requires N_{it} training iterations of $\boldsymbol{u}_\theta$ to get a reliable result for the mean and covariance (i.e. k=1 and k=2 in equation (4), respectively).

When the complexity and/or noise level increases, the total required number of samples $[N_{it} \cdot (N_{MC} \cdot N_{par})]$ will increase quickly. We desire to derive a more efficient approach. Our idea is to approximate $p(\mathbf{u})$ using a variational density and learn the density using Variational Inference (VI).

2.3 Link with Stochastic Partial Differential Equations

As one can note, Stochastic Partial Differential Equations (SPDEs) (equation (5)) are linked with equation (3b).

$$\mathcal{A}_{\boldsymbol{x},\boldsymbol{t}}[\mathbf{u}](\boldsymbol{x},\boldsymbol{t}) = \partial_{\boldsymbol{t}}\mathbf{u}(\boldsymbol{x},\boldsymbol{t}) + \mathcal{A}_{\boldsymbol{x}}[\mathbf{u}](\boldsymbol{x},\boldsymbol{t}) = \boldsymbol{\omega}(\boldsymbol{x},\boldsymbol{t}) \tag{5}$$

with $\mathbf{u}$ is the solution of equation (5), $\boldsymbol{\omega}(\boldsymbol{x},\boldsymbol{t})$ the stochastic force term. $\boldsymbol{\omega}$ can be modelled using a Gaussian Process (GP), with mean $\mu_{\boldsymbol{\omega}}$, and kernel $\sigma^2_{\boldsymbol{\omega}}$.

$$\boldsymbol{\omega} \sim \mathcal{GP}(\mu_{\boldsymbol{\omega}}, \sigma^2_{\boldsymbol{\omega}})$$

In case that $\mathcal{A}_{\boldsymbol{x}}$ is a linear operator, from Rasmussen & Williams (2006) it is known that the function $\mathbf{u}$ is also a GP where the mean and covariance function are governed by the SPDEs.

$$\mathbf{u}(\boldsymbol{x},\boldsymbol{t}) \sim \mathcal{GP}(\mu_{\mathbf{u}}, \sigma^2_{\mathbf{u}})$$

3 A variational approach to Physics Informed Neural Networks

3.1 Bayes' rule

The posterior for the quantity $\mathbf{u}$, given the measurements $\mathbf{y}$, is given by *Bayes' rule* [Bolstad & Curran (2017)]. We assume to have access to the measurement model $p(\mathbf{y},\mathbf{u})$, and prior $p(\mathbf{u})$.

$$p(\mathbf{u}|\mathbf{y}) = \frac{p(\mathbf{y},\mathbf{u})}{p(\mathbf{y})} = \frac{p(\mathbf{y}|\mathbf{u})p(\mathbf{u})}{p(\mathbf{y})} \tag{6}$$

3.1.1 Finite-Dimensional Bayesian inference

Assume that the quantity $\mathbf{u}$ is actually a function, describe by equation (7) where $\boldsymbol{A}$ is non-singular and $\boldsymbol{\omega} \sim \mathcal{GP}(\mu_{\boldsymbol{\omega}}, \Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}})$.

$$\boldsymbol{A}\mathbf{u} = \boldsymbol{\omega} \tag{7}$$

Due to the linearity of equation (7), it is known that [Sarkka & Hartikainen (2012), Lindgren et al. (2011)]

$$\mathbf{u} \sim p(\mathbf{u}) = \mathcal{GP}(\mu_{\mathbf{u}}, \Sigma_{\mathbf{u}\mathbf{u}})$$

where

$$\begin{aligned} \mu_{\mathbf{u}} &= \mathbb{E}[\mathbf{u}] = \mathbb{E}[\boldsymbol{A}^{-1}\boldsymbol{\omega}] = \boldsymbol{A}^{-1}\mu_{\boldsymbol{\omega}} \\ \Sigma_{\mathbf{u}\mathbf{u}} &= \mathbb{E}\left[(\mathbf{u}-\mu_{\mathbf{u}})(\mathbf{u}-\mu_{\mathbf{u}})^T\right] = \mathbb{E}\left[\boldsymbol{A}^{-1}(\boldsymbol{\omega}-\mu_{\boldsymbol{\omega}})(\boldsymbol{\omega}-\mu_{\boldsymbol{\omega}})^T\boldsymbol{A}^{-T}\right] = \boldsymbol{A}^{-1}\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}\boldsymbol{A}^{-T} \end{aligned} \tag{8}$$

Additionally, we assume that a measurement is taken

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{B}\mathbf{u}, \Sigma_{\mathbf{yy}})$$

such that

$$\begin{bmatrix}\mathbf{u}\\ \mathbf{y}\end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix}\mu_{\mathbf{u}}\\ \boldsymbol{B}\mu_{\mathbf{u}}\end{bmatrix}, \begin{bmatrix}\Sigma_{\mathbf{uu}} & \Sigma_{\mathbf{uu}}\boldsymbol{B}^T\\ \boldsymbol{B}\Sigma_{\mathbf{uu}} & \Sigma_{\mathbf{yy}} + \boldsymbol{B}\Sigma_{\mathbf{uu}}\boldsymbol{B}^T\end{bmatrix}\right) \tag{9}$$

To conclude, the conditional probability for the means and variance is described by

$$\begin{aligned}\mu_{\mathbf{u}|\mathbf{y}} &= \mu_{\mathbf{u}} + \Sigma_{\mathbf{uu}}\boldsymbol{B}^T(\Sigma_{\mathbf{yy}} + \boldsymbol{B}\Sigma_{\mathbf{uu}}\boldsymbol{B}^T)^{-1}(\mathbf{y} - \boldsymbol{B}\mu_{\mathbf{u}})\\ \Sigma_{\mathbf{uu}|\mathbf{y}} &= \Sigma_{\mathbf{uu}} - \Sigma_{\mathbf{uu}}\boldsymbol{B}^T(\Sigma_{\mathbf{yy}} + \boldsymbol{B}\Sigma_{\mathbf{uu}}\boldsymbol{B}^T)^{-1}\boldsymbol{B}\Sigma_{\mathbf{uu}}\end{aligned} \tag{10}$$

3.2 Variational Inference for SPDE

Bayes' rule is elegant, but quite often not applicable in practice due to the unknown measurement distribution $p(\mathbf{y})$. A popular technique that is used to approximate the posterior is VI [Ganguly & Earp (2021)]. The idea is to fit a variational density q, which is a member of some density family $\mathcal{Q}$, by solving an optimisation problem. By minimising the Variational Free Energy (VFE) [Wu et al. (2019)], q can be found.

$$q^*(\mathbf{u}) = \arg\min_{q\in\mathcal{Q}} \mathbb{E}_{q(\mathbf{u})}\left[\log\left(\frac{q(\mathbf{u})}{p(\mathbf{y},\mathbf{u})}\right)\right] \tag{11}$$

The quantity $\mathbf{u}$ is actually a function, where the entries of the vector $\mathbf{u}$ can be seen as different function evolutions. The objective is now to find a variational density representing the probability of the function $\mathbf{u}$, that satisfies the SPDEs, and, additionally, conditions the likelihood on the measurements, $\mathbf{y}$.

The discussion will be limited to static SPDEs, though we note that the approach generalises trivially to spatio-temporal SPDEs.

3.2.1 Finite-dimensional Variational Inference

We start reviewing a general finite-dimensional case and comparing it to the results from Bayesian Inference (section 3.1.1), with the objective to obtain the same results as equation (8).

Using a variational approximation, following Opper (2019), this can be found by introducing a variational density

$$\mathbf{u} \sim q(\mathbf{u}) = \mathcal{GP}(\mu_q, \Sigma_{qq})$$

with the free energy is given by

$$\mathcal{L}_{VI}[\mathbf{u}] = \mathbb{E}_{q(\mathbf{u})}\left[\log\left(\frac{q(\mathbf{u})}{p(\mathbf{u})}\right)\right] \tag{12}$$

where

$$\begin{aligned}q(\mathbf{u}) &= |2\pi\Sigma_{qq}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{u}-\mu_q)^T\Sigma_{qq}^{-1}(\mathbf{u}-\mu_q)\right)\\ p(\boldsymbol{\omega}|\mathbf{u}) \equiv p(\mathbf{u}) &= |2\pi\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\boldsymbol{A}\mathbf{u}-\mu_{\boldsymbol{\omega}})^T\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}(\boldsymbol{A}\mathbf{u}-\mu_{\boldsymbol{\omega}})\right)\end{aligned} \tag{13}$$

The VFE can be evaluated to find equation (14). The stepwise calculation can be found in Appendix B.1.

$$\mathcal{L}_{VI}[q] \propto -\frac{1}{2}\log(|\Sigma_{qq}|) + \frac{1}{2}tr(\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}\boldsymbol{A}\Sigma_{qq}\boldsymbol{A}^T) + \frac{1}{2}(\boldsymbol{A}\mu_q - \mu_{\boldsymbol{\omega}})^T\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}(\boldsymbol{A}\mu_q - \mu_{\boldsymbol{\omega}}) \tag{14}$$

To find the minimum of $\mathcal{L}_{VI}[q]$, the first partial derivative with respect to μ_q and Σ_{qq} are set equal to 0. Using Petersen & Pedersen (2012), this results in equation (15). One verifies that equation (8) and equation (15) give the same result.

$$\mu_q = \boldsymbol{A}^{-1}\mu_{\boldsymbol{\omega}} \qquad\qquad \Sigma_{qq} = \boldsymbol{A}^{-1}\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}\boldsymbol{A}^{-T} \tag{15}$$

When adding additional measurements, equation (12) can be extended to

$$\mathcal{L}_{VI}^{\mathbf{y}}[\mathbf{u}] = \mathbb{E}_{q(\mathbf{u})}\left[\log\left(\frac{q(\mathbf{u})}{p(\mathbf{u},\mathbf{y})}\right)\right] \tag{16}$$

and results in

$$\begin{aligned}\mathcal{L}_{VI}^{\mathbf{y}}[q] \propto \underbrace{\frac{1}{2}(\boldsymbol{A}\mu_q - \mu_{\boldsymbol{\omega}})^T\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}(\boldsymbol{A}\mu_q - \mu_{\boldsymbol{\omega}})}_{(17.1)} + \underbrace{\frac{1}{2}tr\left(\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}\boldsymbol{A}\Sigma_{qq}\boldsymbol{A}^T\right)}_{(17.2)} \\ + \underbrace{\frac{1}{2}(\boldsymbol{B}\mu_q - \mathbf{y})^T\Sigma_{\mathbf{y}\mathbf{y}}^{-1}(\boldsymbol{B}\mu_q - \mathbf{y})}_{(17.3)} + \underbrace{\frac{1}{2}tr\left(\Sigma_{\mathbf{y}\mathbf{y}}^{-1}\boldsymbol{B}\Sigma_{qq}\boldsymbol{B}^T\right)}_{(17.4)} - \underbrace{\frac{1}{2}\log|\Sigma_{qq}|}_{(17.5)}\end{aligned} \tag{17}$$

The stepwise calculation can be found in Appendix B.2. One can verify that the minimum of equation (17), found by setting the partial derivatives equal to 0, gives the same results as equation (9).

Once we have access to the mean and covariance, in fact, the job is done, and one can come back to regular inference with GP. There exist standard techniques to find the mean and covariance for given SPDEs, however, all of them depend on linear function decompositions (e.g. Eigenfunctions, Finite Element Method (FEM), Finite Difference Method (FDM), ...) of the function $\mathbf{u}$, and shift all uncertainty to the coefficients. Rather, we would want to draw a connection with the PINN paradigm, specifically by approximating $\mu_{\mathbf{u}}$ and $\sigma_{\mathbf{u}}^2$.

The ambition is to approach an SPDEs similar to how a PINN approaches a PDEs.

3.2.2 Ad-hoc generalisation to infinite-dimensional case

The next step is to establish an ad hoc generalisation to the infinite-dimensional case. This is done using the two analogies between: a vector with infinite vector entries and function evaluations at arbitrary arguments, and matrices and linear operators.

Consider the static SPDEs in equation (18), where $\mathcal{A}_{\boldsymbol{x}}$ is some linear operator and $\boldsymbol{\omega} \sim \mathcal{GP}(\mu_{\boldsymbol{\omega}}, \sigma_{\boldsymbol{\omega}}^2)$.

$$\mathcal{A}_{\boldsymbol{x}}[\mathbf{u}] = \boldsymbol{\omega} \tag{18}$$

Due to the linearity, we have $\mathbf{u} \sim \mathcal{GP}(\mu_{\mathbf{u}}, \sigma_{\mathbf{u}}^2)$. Further assume that we take point measurements $\mathbf{y}$, at spatial coordinates $\boldsymbol{x_m}$, contained in some data set $\mathcal{D}_{DATA}$.

$$\mathbf{y} \sim \mathcal{N}(\mathcal{B}_{\boldsymbol{x}}[\mathbf{u}], \sigma_{\mathbf{y}}^2) \qquad \mathcal{D}_{DATA} = \{(\boldsymbol{x_{m_i}}, \mathbf{y}_i)|i = 1 \dots N_m\}$$

where:

$$\mathcal{B}_{\boldsymbol{x}}[\mathbf{u}] = \mathbf{u}(\boldsymbol{x_m})$$

or

$$\mathcal{B}_{\boldsymbol{x}}[\mathbf{u}] = \int \mathbf{u}(\boldsymbol{x}')\delta(\boldsymbol{x} - \boldsymbol{x}')d\boldsymbol{x}'$$

We already know that

$$\mathbf{u} \sim \mathcal{GP}(\mu_{\mathbf{u}}, \sigma_{\mathbf{u}}^2)$$

Using Opper (2019), this allows us to write the probability of a realisation of $\mathbf{u}$ as

$$-\log p(\mathbf{u}) \propto \exp\Big(-\frac{1}{2}\iint[\mathbf{u}(\boldsymbol{x}) - \mu_{\mathbf{u}}(\boldsymbol{x})]^T\sigma_{\mathbf{u}}^{-2}(\boldsymbol{x},\boldsymbol{x}')[\mathbf{u}(\boldsymbol{x}) - \mu_{\mathbf{u}}(\boldsymbol{x})]d\boldsymbol{x}d\boldsymbol{x}'\Big) \tag{19}$$

and the probability of the measurement data is written as

$$p(\mathcal{D}_{DATA}|\mathbf{u}) = \prod_{(\boldsymbol{x_m},\mathbf{y})\in\mathcal{D}_{DATA}} ||2\pi\sigma_{\mathbf{y}}^2||^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\sigma_{\mathbf{y}}^{-2}||\mathbf{y} - \mathcal{B}_{\boldsymbol{x}}[\mathbf{u}]||^2\right) \tag{20}$$

Next, the ad-hoc generalisation of equation (17) is constructed.

$$(17.1) \Rightarrow \frac{1}{2} \iint_{\mathcal{D}^2_{PHYS}} \mathcal{A}_{\boldsymbol{x}}[\mu_q]^T \sigma_{\boldsymbol{\omega}}^{-2}(\boldsymbol{x}, \boldsymbol{x}') \mathcal{A}_{\boldsymbol{x}}[\mu_q] d\boldsymbol{x} d\boldsymbol{x}' \tag{21}$$

$$(17.2) \Rightarrow \frac{1}{2} \iint_{\mathcal{D}^2_{PHYS}} tr\left(\sigma_{\boldsymbol{\omega}}^{-2} \mathcal{A}_{\boldsymbol{x}}[\sigma_q^2(\boldsymbol{x}, \boldsymbol{x}')] \mathcal{A}_{\boldsymbol{x}'}^T\right) d\boldsymbol{x} d\boldsymbol{x}' \tag{22}$$

$$(17.3) \Rightarrow \frac{1}{2} \sum_{(\boldsymbol{x_m}, \mathbf{y}) \in \mathcal{D}_{DATA}} \sigma_{\mathbf{y}}^{-2} ||\mathbf{y} - \mathcal{B}_{\boldsymbol{x}}[\mu_q]||^2 \tag{23}$$

$$(17.4) \Rightarrow \frac{1}{2} \sum_{(\boldsymbol{x_m}, \mathbf{y}) \in \mathcal{D}_{DATA}} tr\left(\sigma_{\mathbf{y}}^{-2} \mathcal{B}_{\boldsymbol{x}}[\sigma_q^2(\boldsymbol{x_m}, \boldsymbol{x}'_{\boldsymbol{m}})] \mathcal{B}_{\boldsymbol{x}'}^T\right) \tag{24}$$

For (17.5), an ad-hoc solution is a great challenge. We proposed that it can be seen as a regularisation term for σ_q^2. If σ_q^2 were a matrix, then would it be $\log(\det[\sigma_q^2])$. Here σ_q^2 is a covariance kernel, the generalised determinant, $\det[.]$, it is then basically the product of the eigenvalues of the kernel. In practice, only a finite number of query points are being evaluated. The determinant of the associated covariance matrix can be approximated by

$$(17.5) \Rightarrow \det\left[\sigma_q^2\right] \approx \det\left(\Sigma_{qq}^{PHYS}\right) \tag{25}$$

where $\left[\Sigma_{qq}^{PHYS}\right]_{ij} = \sigma_q^2(\boldsymbol{x}_i, \boldsymbol{x}_j); \quad \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_{PHYS}$

The last obstacles in the derivation are the double integrals. Those are impossible to evaluate for arbitrary networks. Hence, the double integral can be approximated by drawing random samples in the spatial dimension.

$$\iint_{\mathcal{D}^2} f(\boldsymbol{x}, \boldsymbol{x}') d\boldsymbol{x} d\boldsymbol{x}' \approx \Delta \boldsymbol{x}^2 \sum_{(\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{D}^2} f(\boldsymbol{x}, \boldsymbol{x}') \tag{26}$$

Filling equation (21-26)into equation (17), the approximated total free energy can be calculated

$$\mathcal{L}_{VI}^{\mathbf{y}}[\mu_q, \sigma_q^2] \approx \underbrace{\frac{1}{2} \Delta \boldsymbol{x}^2 \sum_{(\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{D}^2_{PHYS}} \mathcal{A}_{\boldsymbol{x}}[\mu_q]^T \sigma_{\boldsymbol{\omega}}^{-2}(\boldsymbol{x}, \boldsymbol{x}') \mathcal{A}_{\boldsymbol{x}'}[\mu_q]}_{J_1[\mu_q]} + \underbrace{\frac{1}{2} \Delta \boldsymbol{x}^2 \sum_{(\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{D}^2_{PHYS}} tr\left(\sigma_{\boldsymbol{\omega}}^{-2} \mathcal{A}_{\boldsymbol{x}}[\sigma_q^2(\boldsymbol{x}, \boldsymbol{x}')] \mathcal{A}_{\boldsymbol{x}'}^T\right)}_{J_2[\sigma_q^2]}$$
$$+ \underbrace{\frac{1}{2} \sum_{(\boldsymbol{x_m}, \mathbf{y}) \in \mathcal{D}_{DATA}} tr\left(\sigma_{\mathbf{y}}^{-2} \mathcal{B}_{\boldsymbol{x}}[\sigma_q^2(\boldsymbol{x_m}, \boldsymbol{x_m})] \mathcal{B}_{\boldsymbol{x}}^T\right)}_{J_3[\sigma_q^2]} + \underbrace{\frac{1}{2} \sum_{(\boldsymbol{x_m}, \mathbf{y}) \in \mathcal{D}_{DATA}} \sigma_{\mathbf{y}}^{-2} ||\mathbf{y} - \mathcal{B}_{\boldsymbol{x}}[\mu_q]||^2}_{J_4[\mu_q]} - \underbrace{\frac{1}{2} \log \left|\det\left[\Sigma_{qq}^{PHYS}\right]\right|}_{J_5[\sigma_q^2]} \tag{27}$$

3.3 Relation with Physics Informed Neural Network

Equation (27) can be split into two parts. A part that is similar to a standard PINN cost function (equation (2) w.r.t. μ_q).

$$J_1[\mu_q] + J_4[\mu_q] = \frac{1}{2} \Delta \boldsymbol{x}^2 \sum_{(\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{D}^2_{PHYS}} \mathcal{A}_{\boldsymbol{x}}[\mu_q](\boldsymbol{x})^T \sigma_{\boldsymbol{\omega}}^{-2}(\boldsymbol{x}, \boldsymbol{x}') \mathcal{A}_{\boldsymbol{x}'}[\mu_q](\boldsymbol{x}') + \frac{1}{2} \sum_{(\boldsymbol{x_m}, \mathbf{y}) \in \mathcal{D}_{DATA}} \sigma_{\mathbf{y}}^{-2} ||\mathbf{y} - \mathcal{B}_{\boldsymbol{x}}[\mu_q]||^2$$

and a second part that expresses the uncertainty (w.r.t. σ_q^2).

$$J_2[\sigma_q^2] - J_5[\sigma_q^2] + J_3[\sigma_q^2] = \frac{1}{2} \Delta \boldsymbol{x}^2 \sum_{(\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{D}^2_{PHYS}} tr\left(\sigma_{\boldsymbol{\omega}}^{-2} \mathcal{A}_{\boldsymbol{x}}[\sigma_q^2(\boldsymbol{x_m}, \boldsymbol{x}'_{\boldsymbol{m}})] \mathcal{A}_{\boldsymbol{x}'}^T\right) - \frac{1}{2} \log \left|\det\left[\Sigma_{qq}^{PHYS}\right]\right|$$
$$+ \frac{1}{2} \sum_{(\boldsymbol{x_m}, \mathbf{y}) \in \mathcal{D}_{DATA}} tr\left(\sigma_{\mathbf{y}}^{-2} \mathcal{B}_{\boldsymbol{x}}[\sigma_q^2(\boldsymbol{x_m}, \boldsymbol{x}'_{\boldsymbol{m}})] \mathcal{B}_{\boldsymbol{x}}^T\right) \tag{28}$$

As one can see, there is no explicit tuning factor, λ. The tuning results from $\sigma_{\mathbf{y}}^2$, $\sigma_{\boldsymbol{\omega}}^2$ and $\Delta\boldsymbol{x}$, i.e. the spatial coarseness.

4 RESULTS

4.1 PROBLEM FORMULATION

To validate our proposed framework, a 1D steady state Disturbed Thermal Field (DTF) is examined, which can be described by means of the following SPDEs [Fourier (1878)] and Boundary Condition (BC):

$$k\frac{\partial^2 \mathbf{u}(\boldsymbol{x})}{\partial \boldsymbol{x}^2} = \boldsymbol{\omega}(\boldsymbol{x}) \qquad \text{BC}: \ \mathbf{u}(0) = \mathbf{u}(1) = 1 \tag{29}$$

with $\boldsymbol{x} \in [0, 1.0]$, the spatial coordinates, $k > 0$, the conductivity constant, $\boldsymbol{\omega} \sim \mathcal{GP}(\mu_{\boldsymbol{\omega}}, \sigma_{\boldsymbol{\omega}}^2)$, the stochastic heat flux term and $\mathbf{u} \sim \mathcal{GP}(\mu_{\mathbf{u}}, \sigma_{\mathbf{u}}^2)$, the stochastic temperate field. Take a 1D steel beam with a length of 1 m with homogeneous material properties. Over the length of the beam some torches are placed, as shown in Figure 2. Normalizing equation (29), such that k = 1, results in a DTF with $\mu_{\boldsymbol{\omega}} = (6\pi)^2 \sin(6\pi \cdot \boldsymbol{x})$, and variance, $\sigma_{\boldsymbol{\omega}}^2 = 4,0 \cdot \delta(\boldsymbol{x} - \boldsymbol{x}')$.

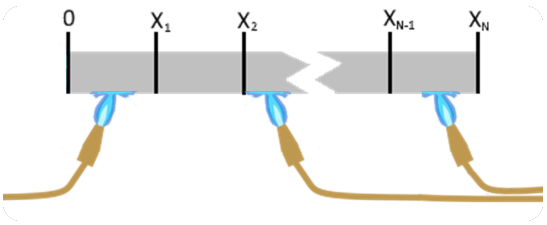


Figure 2: 1D steel beam with distributed heat sources

For the training of the different NNs, 25 $\boldsymbol{x_m}$-points are chosen evenly spread over $[0, 0.6]$. For the testing/validation, 200 $\boldsymbol{x}$-points are chosen evenly spread over $[0, 1.0]$. This allows us to compare the inter-/and extrapolation capabilities of the B-PINN and VI-PINN. The training points have some noise $\epsilon_{\mathbf{y}} \sim \mathcal{GP}(0, 1.5^2)$, added to emulate measurement noise.

It was found empirically that rescaling the input $\boldsymbol{x}$, and measurement data $\mathbf{y}$, yielded better results and reduced computation time.

$$\mathcal{D}_{DATA} \in \mathbb{R}^n \mapsto \mathcal{D'}_{DATA} \in [0, 1]^n$$

4.2 SOLVE WITH A B-PINN USING NUTS

The implementation of the HMC technique is performed using the NUTS [Hoffman & Gelman (2011)] implementation in Python using NumPyro [Phan et al. (2019), Bingham et al. (2019)]. This algorithm is an extension of HMC. It adaptively sets the step-size, lr, and the number of samples, N_{HMC}, which reduces the computation time. In Table 1, four combinations of the following hyperparameters are displayed:

- ϵ : The number of training iterations.
- $BuIn$: The number of burn-in steps.
- lr_0 : The initial step-size or learning rate.
- Layers : The architecture of the network.
- TAP : The Target Acceptance Probability.
- N_{PDEs} : The Number of points used to evaluate the PDEs.
- Key : The random key used for the random generator.

For all the hyperparameter configurations, the maximum number of samples, N_{HMC}, is 1023. In Figure 3, the corresponding results are shown.

Table 1: Hyperparameters NUTS-B-PINN.

It	ϵ	$BuIn$	lr_0	Layers	TAP	N_{PDEs}	Key
(a)	$5e^4$	$5e^2$	$1e^{-4}$	[1, 3, 16, 20, 32, 20, 16, 3, 1]	0,80	1000	1234
(b)	$5e^4$	$5e^2$	$5e^{-4}$	[1, 3, 10, 16, 20, 32, 25, 20, 16, 10, 3, 1]	0,85	1200	12345
(c)	$5e^3$	$5e^1$	$1e^{-5}$	[1, 3, 10, 16, 20, 32, 20, 16, 10, 3, 1]	0,80	1000	20901
(d)	$5e^4$	$5e^2$	$1e^{-4}$	[1, 3, 16, 25, 16, 3, 1]	0,80	800	1234

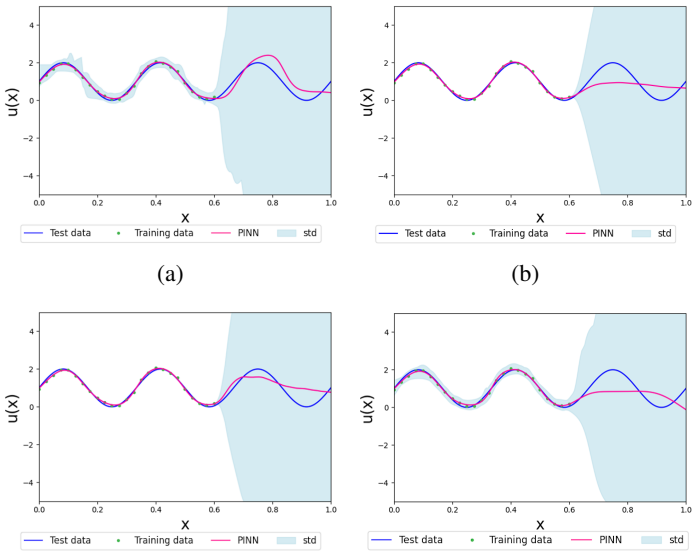


Figure 3: Results NUTS-B-PINN

4.3 Solve with a VI-PINN

The general architecture of VI-PINN is shown in Figure 4. The mean μ_q is estimated with a NN. For the variance σ_q^2, the following the kernel is proposed

$$\sigma_q^2 = \sigma_{NN}^2 \exp\left[\alpha||\boldsymbol{x}_i - \boldsymbol{x}_j||^2\right]$$

with $\sigma_{NN}^2 = ||\sigma_{NN}(\boldsymbol{x}_i)\sigma_{NN}(\boldsymbol{x}_j)||$ where $\sigma_{NN}(\boldsymbol{x})$ is a NN with input $\boldsymbol{x}$. The choice is made to take the product instead of a NN with two inputs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$, to preserve the symmetry of the variance $(\sigma_{NN}^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_{NN}^2(\boldsymbol{x}_j, \boldsymbol{x}_i))$. For this problem $\mathcal{A}_{\boldsymbol{x}}[\mu_q]$ and $\mathcal{A}_{\boldsymbol{x}}[\sigma_q^2(\boldsymbol{x}, \boldsymbol{x}')]\mathcal{A}_{\boldsymbol{x}'}^T$ are given by

$$\mathcal{A}_{\boldsymbol{x}}[\mu_q] \equiv \bigtriangledown_{\boldsymbol{x}}^2 \mu_q - \mu_{\boldsymbol{\omega}}$$
$$\mathcal{A}_{\boldsymbol{x}}[\sigma_q^2(\boldsymbol{x}, \boldsymbol{x}')]\mathcal{A}_{\boldsymbol{x}'}^T \equiv \bigtriangledown_{\boldsymbol{x}}^2 \bigtriangledown_{\boldsymbol{x}'}^2 \sigma_q^2(\boldsymbol{x}, \boldsymbol{x}') - \sigma_{\boldsymbol{\omega}}^2 \delta(\boldsymbol{x} - \boldsymbol{x}')$$

To reduce the training time, $\bigtriangledown_{\boldsymbol{x}}^2 \bigtriangledown_{\boldsymbol{x}'}^2 \sigma_q^2(\boldsymbol{x}, \boldsymbol{x}')$ is calculated explicitly for this 1D problem.

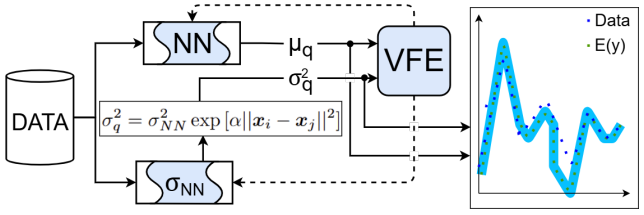
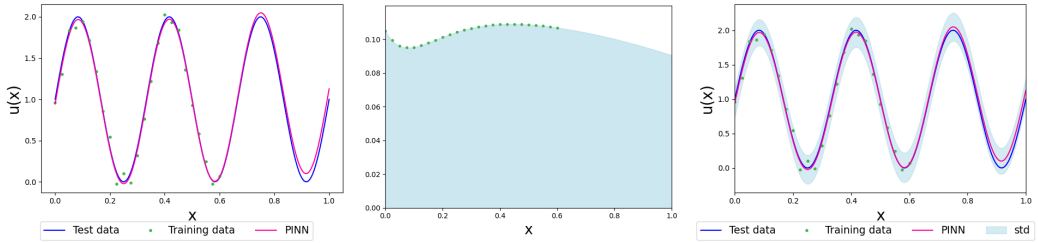


Figure 4: General architecture of a VI-PINN

The estimated mean μ_q, and variance σ_q^2, are evaluated in equation (27) to find the VFE. In Table 2, the hyperparameters of the NNs for μ_q and σ_{NN}, used in the VI-PINN-implementation, are shown. The intermediate results for the mean μ_q, and the standard deviation σ_q, are shown respectively in Figure 5a and Figure 5b. In Figure 5c, the combined result is shown.

Table 2: Hyper parameters of the 2 NN for the VI-PINN.

NN_{μ_q}	$\text{NN}_{\sigma_{NN}}$	lr	ϵ_{μ_q}	$\epsilon_{\sigma_{NN}}$	N_{PDEs}	$\sigma_{\boldsymbol{\omega}}$	σ_y	α
[1, 16, 20, 16, 1]	[1, 5, 8, 12, 12, 12, 10, 1]	$1e^{-4}$	$15e^3$	$10e^3$	700	2,0	1,5	500



(a) The mean μ_q (b) The standard deviation σ_q (c) Combined results

Figure 5: Results VI-PINN

4.4 Discuss the results

Table 3: Results from specify solving techniques

Solving technique	Training time*	MSE	AR
NUTS-B-PINN (a)	23 min 10,93 s	$7,93e^{-2}$	$100,00\%$
NUTS-B-PINN (b)	55 min 57,72 s	$1,48e^{-1}$	$41,00\%$
NUTS-B-PINN (c)	1 min 57,36 s	$9,03e^{-2}$	$41,50\%$
NUTS-B-PINN (d)	8 min 49,34 s	$1,76e^{-1}$	$100,00\%$
VI-PINN	4 min 39,57 s	$6,97e^{-6}$	$100,00\%$

*All calculations are performed on 32 CPUs (Intel(R) Core (TM) i9-14900K)

In Table 3, the training time, Means Square Error (MSE), and Acceptance Rate (AR) of each solving technique are shown.

4.4.1 Mean estimation

The two techniques both provide a good estimation for μ_q inside the measurement range ($\boldsymbol{x} \in [0, 0.6]$). When looking outside the measurement range, there is a notable difference. VI-PINN can extrapolate and can yield a good solution. B-PINN can not, even when doing more iterations.

4.4.2 Variance estimation

For B-PINN, the variance (and standard deviation) shows a rising behaviour. Outside the measurement range ($\boldsymbol{x} \in [0.6, 1]$), the σ_q estimation exploits to high values (>10).

When comparing to the VI-PINN, both have a similar result inside the measurement range ($\boldsymbol{x} \in [0, 0.6]$). When looking outside the measurement range ($\boldsymbol{x} \in [0.6, 1]$), a noticeable difference can be observed. The σ_q estimation of the VI-PINN estimation stays bounded.

This can be explained because the variance σ_q^2 is included explicitly in the government equation of VI-PINN (see equation (28)) and in that way can be regulated. The B-PINN estimation is sample-based, hence has no explicit way of regulating it.

4.4.3 COMBINED RESULT (MEAN AND STANDARD DEVIATION)

Examining the combined result of the mean μ_q and standard deviation σ_q shows that the results from the VI-PINN are significantly better than those of the B-PINN, especially when extrapolating. This expresses itself in a low MSE and high AR, compared to the B-PINN estimations.

4.4.4 TRAINING TIME

The training time of the B-PINN is case-dependent. When the results after an iteration have a high uncertainty, more samples are needed, and the iteration will take longer. A slight change in a hyperparameter can result in a significant increase in training time. Increasing the number of iterations does not always result in better estimations.

VI-PINN don't rely on sampling, therefore it's less sensitive to changes in hyperparameters in terms of training time, which makes it easier to tune than B-PINN, which results in a lower training time.

5 CONCLUSION

PINN has been shown to give good results, even with small datasets. It does so by extending the NN framework with physics, represented using PDEs or ODEs. But it has its limitations. PINNs has no uncertainty quantification built into it. In the literature, a B-PINN is proposed to solve this limitation. This is a sample-based implementation, where all parameters are assumed stochastic.

We proposed the VI-PINN as a new way of including uncertainty estimation into the PINN predictions. The VI-PINN achieves this due to its strong mathematical backbone, where VI is used to estimate the posterior in combination with VFE, which is used as a cost function in this probabilistic framework.

Comparing the B-PINN and VI-PINN, both yield an acceptable result inside the training range. Looking at the extrapolating capabilities, B-PINN falls short, a gap that VI-PINN successfully fills. The physics is firmly embedded into VI-PINN, both in the estimation of μ_q and σ_q^2. This gives it the ability to give reliable results even when extrapolating. The cost function equation (27) for VI-PINN is an explicit function and therefore easier to implement and evaluate than B-PINN. This results in faster training times and results that better align with reality over the full range.

But it still has some flaws, such as a long training time and being unable to include implicit BC.

To conclude this paper, we achieve our ambition of approaching the SPDEs similarly to how PINNs approach PDEs, while the uncertainty is moved away from the parameters.

ACKNOWLEDGMENTS THE FUNDING

This work was realised through the financial support of the Flanders Make research project: SBO-FM-DTF-PINN.

CONFLICT OF INTEREST

The authors declare no potential conflict of interest.

REFERENCES

Saakaar Bhatnagar, Andrew Comerford, and Araz Banaeizadeh. Physics Informed Neural Networks for Modeling of 3D Flow-Thermal Problems with Sparse Domain Data. *Journal of Machine Learning for Modeling and Computing*, 5, 01 2024. doi: 10.1615/JMachLearnModelComput.2024051540.

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *J. Mach. Learn. Res.*, 20:28:1–28:6, 2019. URL `http://jmlr.org/papers/v20/18-403.html`.

William M. Bolstad and James M. Curran. Introduction to Bayesian Statistics. In *Logic, Probability, and Uncertainty*. New York: Wiley, 2017. 3rd ed.

Adam D. Cobb and Brian Jalaian. Scaling Hamiltonian Monte Carlo inference for Bayesian Neural Networks with symmetric splitting. In Cassio de Campos and Marloes H. Maathuis (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 675–685. PMLR, 27–30 Jul 2021. URL https://proceedings.mlr.press/v161/cobb21a.html.

J. Fourier. The Analytical Theory of Heat. In *The Analytical Theory of Heat*. The Univeresity, 1878.

Eric Fowler, Christopher J. McDevitt, and Subrata Roy. Physics-Informed Neural Network simulation of thermal cavity flow. *Scientific Reports*, 14(1):15203, Jul 2024. ISSN 2045-2322. doi: 10.1038/s41598-024-65664-3. URL https://doi.org/10.1038/s41598-024-65664-3.

Ankush Ganguly and Samuel W. F. Earp. An Introduction to Variational Inference, 2021. URL https://arxiv.org/abs/2108.13083.

Matthew D. Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo, 2011. URL https://arxiv.org/abs/1111.4246.

Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-On Bayesian Neural Networks—A Tutorial for Deep Learning Users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, May 2022. ISSN 1556-6048. doi: 10.1109/mci.2022.3155327. URL http://dx.doi.org/10.1109/MCI.2022.3155327.

Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between Gaussian fields and Gaussian Markov random fields: the Stochastic Partial Differential Equation approach. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 73(4):423–498, 2011.

R.S.H. Mah and V. Chakravarthy. Pattern recognition using artificial Neural Networks. *Computers & Chemical Engineering*, 16(4):371–377, 1992. ISSN 0098-1354. doi: https://doi.org/10.1016/0098-1354(92)80054-D. URL https://www.sciencedirect.com/science/article/pii/009813549280054D. Neutral network applications in chemical engineering.

Stefano Markidis. The Old and the New: Can Physics-Informed Deep-Learning Replace TraditionalLinear Solvers? *Frontiers in Big Data*, Volume 4 - 2021, 2021. ISSN 2624-909X. doi: 10.3389/fdata.2021.669097. URL https://www.frontiersin.org/journals/big-data/articles/10.3389/fdata.2021.669097.

Manfred Opper. Variational Inference for Stochastic Differential Equations. *Annalen der Physik*, 531(3):1800233, 2019.

K. B. Petersen and M. S. Pedersen. The Matrix Cookbook, nov 2012. URL http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html. Version 20121115.

Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro. *arXiv preprint arXiv:1912.11554*, 2019.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics Informed Deep Learning (part i): Data-driven Solutions of Nonlinear Partial Differential Equations, 2017. URL https://arxiv.org/abs/1711.10561.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006. ISBN 9780262182539.

Simo Sarkka and Jouni Hartikainen. Infinite-dimensional Kalman Filtering approach to spatio-temporal Gaussian Process regression. In *Artificial intelligence and statistics*, pp. 993–1001. PMLR, 2012.

Dian Wu, Lei Wang, and Pan Zhang. Solving Statistical Mechanics Using Variational Autoregressive Networks. *Phys. Rev. Lett.*, 122:080602, Feb 2019. doi: 10.1103/PhysRevLett.122.080602. URL https://link.aps.org/doi/10.1103/PhysRevLett.122.080602.

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

A ABBREVIATION

AR	Acceptance Rate
BC	Boundary Condition
BNN	Bayesian Neural Network
B-PINN	Bayesian PINN
DTF	Disturbed Thermal Field
FDM	Finite Difference Method
FEM	Finite Element Method
GP	Gaussian Process
HMC	Hamiltonian Monte Carlo
NN	Neural Network
NUTS	No-U-Turn Sampler
MSE	Means Square Error
ODEs	Ordinary Differential Equations
PDEs	Partial Differential Equations
PINN	Physics-Informed Neural Network
SPDEs	Stochastic Partial Differential Equations
VFE	Variational Free Energy
VI	Variational Inference
VI-PINN	Variational Inference Physics Informed Neural Network

B CALCULATION OF THE TERMS OF THE VARIATIONAL FREE ENERGY

B.1 FREE ENERGY WITHOUT MEASUREMENTS

Given:

$$\mathcal{L}_{VI}[q] = \mathbb{E}_{q(\mathbf{u})}\left[\log\left(\frac{q(\mathbf{u})}{p(\mathbf{u})}\right)\right]$$

Calculate $\log\left(\frac{q(\mathbf{u})}{p(\mathbf{u})}\right)$, using

$$q(\mathbf{u}) = |2\pi\Sigma_{qq}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{u}-\mu_q)^T\Sigma_{qq}^{-1}(\mathbf{u}-\mu_q)\right)$$

$$p(\boldsymbol{\omega}|\mathbf{u}) \equiv p(\mathbf{u}) = |2\pi\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{A}\mathbf{u}-\mu_{\boldsymbol{\omega}})^T\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}(\boldsymbol{A}\mathbf{u}-\mu_{\boldsymbol{\omega}})\right)$$

gives

$$\begin{aligned}\log\left(\frac{q(\mathbf{u})}{p(\mathbf{u})}\right) = &-\frac{1}{2}\left[\log(2\pi) + \log(|\Sigma_{qq}|) - \log(2\pi) - \log(|\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}|)\right] \\ &+\frac{1}{2}\left[(\boldsymbol{A}\mathbf{u}-\mu_{\boldsymbol{\omega}})^T\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}(\boldsymbol{A}\mathbf{u}-\mu_{\boldsymbol{\omega}}) - (\mathbf{u}-\mu_q)^T\Sigma_{qq}^{-1}(\mathbf{u}-\mu_q)\right]\end{aligned}$$

On the first line, the $\log(2\pi)$ can be crossed out, and the other Σ's can be taken together. Take now the expected value $\mathbb{E}_{q(\mathbf{u})} \equiv \mathbb{E}_q$, using Petersen & Pedersen (2012).

$$\mathcal{L}_{VI}[q] = -\frac{1}{2}\underbrace{\mathbb{E}_q\left[\log\left(\frac{|\Sigma_{qq}|}{|\Sigma_{\boldsymbol{\omega}}|}\right)\right]}_{(1)} - \frac{1}{2}\underbrace{\mathbb{E}_q\left[(\mathbf{u}-\mu_q)^T\Sigma_{qq}^{-1}(\mathbf{u}-\mu_q)\right]}_{(2)} + \frac{1}{2}\underbrace{\mathbb{E}_q\left[(\boldsymbol{A}\mathbf{u}-\mu_{\boldsymbol{\omega}})^T\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}(\boldsymbol{A}\mathbf{u}-\mu_{\boldsymbol{\omega}})\right]}_{(3)}$$

$$(1) = \log(|\Sigma_{qq}|) + \underbrace{\log(|\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}|)}_{\text{IND of } q(\mathbf{u})\rightarrow Cst.}$$

Using Equation (328) from Petersen & Pedersen (2012) to solve (2)

$$\mathbb{E}[\boldsymbol{c}^T\boldsymbol{M}\boldsymbol{c}] = tr(\boldsymbol{M}\cdot cov(\boldsymbol{c})) + \mu_{\boldsymbol{c}}^T\boldsymbol{M}\mu_{\boldsymbol{c}}$$

with

$$\boldsymbol{c} = (\mathbf{u}-\mu_q) \text{ and } \boldsymbol{M} = \Sigma_{qq}^{-1}$$

so

$$\mu_{\boldsymbol{c}} = 0 \text{ and } cov(\boldsymbol{c}) = \Sigma_{qq}$$

This results in

$$(2) = tr(\Sigma_{qq}^{-1}\Sigma_{qq}) = 1 \rightarrow Cst.$$

Use the same formula to solve (3), the result in

$$(3) = tr(\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}\boldsymbol{A}\Sigma_{qq}\boldsymbol{A}^T) + (\boldsymbol{A}\mu_q-\mu_{\boldsymbol{\omega}})^T\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}(\boldsymbol{A}\mu_q-\mu_{\boldsymbol{\omega}})$$

Filling in everything, equation (14) is found

$$\mathcal{L}_{VI}[q] = -\frac{1}{2}\log(|\Sigma_{qq}|) + \frac{1}{2}tr(\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}\boldsymbol{A}\Sigma_{qq}\boldsymbol{A}^T) + \frac{1}{2}(\boldsymbol{A}\mu_q-\mu_{\boldsymbol{\omega}})^T\Sigma_{\boldsymbol{\omega}\boldsymbol{\omega}}^{-1}(\boldsymbol{A}\mu_q-\mu_{\boldsymbol{\omega}}) + Cst.$$

B.2 Free energy with measurements

Given

$$\mathcal{L}_{VI}^{\mathbf{y}}[q] = \mathbb{E}_{q(\mathbf{u})}\left[\log\left(\frac{q(\mathbf{u})}{p(\mathbf{u},\mathbf{y})}\right)\right] = \mathbb{E}_{q(\mathbf{u})}\left[\log\left(\frac{q(\mathbf{u})}{p(\mathbf{u})p(\mathbf{y}|\mathbf{u})}\right)\right]$$
$$= \mathbb{E}_{q(\mathbf{u})}\left[\log\left(\frac{q(\mathbf{u})}{p(\mathbf{u})}\right)\right] - \mathbb{E}_{q(\mathbf{u})}\left[\log\left(p(\mathbf{y}|\mathbf{u})\right)\right]$$

$\mathbb{E}_{q(\mathbf{u})}\left[\log\left(\frac{q(\mathbf{u})}{p(\boldsymbol{\omega})}\right)\right]$ is know from Appendix B.1, so only the $\mathbb{E}_{q(\mathbf{u})}\left[\log\left(p(\mathbf{y})\right)\right]$ term must be determinate. Due to linearity of $\boldsymbol{B}$, we know that

$$p(\mathbf{y}|\mathbf{u}) = |2\pi\Sigma_{\mathbf{y}\mathbf{y}}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\boldsymbol{B}\mathbf{u}-\mu_{\mathbf{y}})^T\Sigma_{\mathbf{y}\mathbf{y}}^{-1}(\boldsymbol{B}\mathbf{u}-\mu_{\mathbf{y}})\right)$$

so

$$\log(p(\mathbf{y}|\mathbf{u})) = -\frac{1}{2}(\boldsymbol{B}\mathbf{u}-\mu_{\mathbf{y}})^T\Sigma_{\mathbf{y}\mathbf{y}}^{-1}(\boldsymbol{B}\mathbf{u}-\mu_{\mathbf{y}}) + Cst.$$

Using Equation (328) from Petersen & Pedersen (2012) now for solving $\mathbb{E}_{q(\mathbf{u})}\left[\log\left(p(\mathbf{y}|\mathbf{u})\right)\right]$

$$\mathbb{E}[\boldsymbol{c}^T\boldsymbol{M}\boldsymbol{c}] = tr(\boldsymbol{M}\cdot cov(\boldsymbol{c})) + \mu_{\boldsymbol{c}}^T\boldsymbol{M}\mu_{\boldsymbol{c}}$$

with

$$\boldsymbol{c} = (\boldsymbol{B}\mathbf{u}-\mu_{\mathbf{y}}) \text{ and } \boldsymbol{M} = \Sigma_{\mathbf{y}\mathbf{y}}^{-1}$$

so

$$\mu_{\boldsymbol{c}} = (\boldsymbol{B}\mu_q-\mu_{\mathbf{y}}) \text{ and } cov(\boldsymbol{c}) = \boldsymbol{B}\Sigma_{qq}\boldsymbol{B}^T$$

This results in

$$-\mathbb{E}_{q(\mathbf{u})}\left[\log\left(p(\mathbf{y}|\mathbf{u})\right)\right] = \frac{1}{2}tr(\Sigma_{\mathbf{y}\mathbf{y}}^{-1}\boldsymbol{B}\Sigma_{qq}\boldsymbol{B}^T) + \frac{1}{2}(\boldsymbol{B}\mu_q-\mu_{\mathbf{y}})^T\Sigma_{\mathbf{y}\mathbf{y}}^{-1}(\boldsymbol{B}\mu_q-\mu_{\mathbf{y}}) + Cst.$$