A Bayesian neural network approach to Multi-fidelity surrogate modelling

Baptiste Kerleguer, Claire Cannamela, Josselin Garnier

A Bayesian neural network approach to Multi-fidelity surrogate modelling

Baptiste Kerleguer[1,2], Claire Cannamela[1], and Josselin Garnier[2]

[1]CEA, DAM, DIF, F-91297, Arpajon, France
[2]Centre de Mathématiques Appliquées, Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau Cedex, France
baptiste.kerleguer@polytechnique.edu

Abstract

This paper deals with surrogate modelling of a computer code output in a multi-fidelity context, i.e., when the output can be evaluated at different levels of accuracy and computational cost. Using observations of the output at low- and high-fidelity levels, we propose a method that combines Gaussian process (GP) regression and Bayesian neural network (BNN), in a method called GPBNN. The low-fidelity output is treated as a single-fidelity code using classical GP regression. The high-fidelity output is approximated by a BNN that incorporates, in addition to the high-fidelity observations, well-chosen realisations of the low-fidelity output emulator. The predictive uncertainty of the final surrogate model is then quantified by a complete characterisation of the uncertainties of the different models and their interaction. GPBNN is compared with most of the multi-fidelity regression methods allowing to quantify the prediction uncertainty.

Keywords— Multi-fidelity, Surrogate modelling, Bayesian Neural Network, Gaussian Process Regression

1 Introduction

We consider the situation in which two levels of code that simulate the same system have different costs and accuracies. We would like to build a surrogate model of the most accurate and most costly code level, also called high-fidelity code. The underlying motivation is to carry out an uncertainty propagation study or a sensitivity analysis that require many calls and therefore, the substitution of the high-fidelity code by a surrogate model with quantified prediction uncertainty is necessary. To build the surrogate, a small number N_H of high-fidelity code outputs and a large number N_L of low-fidelity code outputs are given. In some applications we may have $N_L \gg N_H$, but this article will focus on $N_L > N_H$ and the context of small data. Then, the low-fidelity surrogate model uncertainty must be taken into account.

A well-known method to build a surrogate model with uncertainty quantification is Gaussian process (GP) regression, GP 1F in this paper. This method has become popular in computer experiments [26, 23] and now allows scaling up in the number of learning points [22]. The emergence of multi-fidelity codes (codes that can be run at different levels of accuracy and cost) has motivated the introduction of new GP regression approaches. The first one was the Gaussian process auto-regressive or AR(1) scheme proposed by [10]. The form of the AR(1) model expresses a simple and linearity relationships between the codes and it follows from Markov properties given [18]. This method is used in [4] for optimisation. This approach has been improved by [12] with the decoupling of the estimation of the hyper-parameters of the different code levels. In the following the recursive AR(1) model is called the AR(1) model because the results do not change only the computation time is reduced. The Deep GP method introduced in [19] makes it possible to adapt the approach to cases in which the relationships between the code levels are nonlinear. An improvement has been further made by adding to the covariance function of the high-fidelity GP proposed by [19] a linear kernel in [3]. Multi-fidelity GP regression has been used in several fields as illustrated in [24, 21]. Polynomial chaos can also be exploited as in [17] for multi-fidelity regression.

Recent improvements in the implementation of neural networks have motivated research on multi-fidelity neural networks [13]. In [16], the authors combine a fully connected neural network (NN) and a linear system for the interactions between codes. The low-fidelity surrogate model is built using a fully connected NN, see [27] for a direct application. In order to quantify the prediction deviations and to evaluate the reliability of the prediction the NNs have been improved to become Bayesian Neural Networks (BNN) [14]. The multi-fidelity model has been improved by using BNN for high-fidelity modelling in [15]. In our article this

method will be called MBK. The method in [15] offers options for a single-fidelity active learning and is more general for multi-fidelity modelling. The disadvantages of methods using NN are the non-prediction of the model uncertainties and the difficult optimisation of the hyper-parameters in a small data context. These disadvantages can be overcome by the use of BNNs. The ability of BNN for uncertainty quantification is explained in [9].

The purpose of our paper is to present a method competing in terms of prediction with the methods of multi-fidelity regression : AR(1) model, DeepGP and MBK. Our paper also aims to improve the quantification of uncertainties in the non-linear case for all these methods. As not all the proposed methods give Gaussian processes as output, the uncertainties have to be compared differently. An approach, involving two quantities, is proposed in our article. We will use the Gauss-Hermite quadrature to transfer the low-fidelity a posterior law to the high-fidelity BNN. We will compare the properties and results of our strategy with the ones of these methods. For evaluation, we examine our surrogate model approach with two-level multi-fidelity benchmark functions and with a simulation example. The results demonstrate that the method presented in our paper is easier to train and more accurate than any other multi-fidelity neural network-based method. Moreover, the method is more flexible compared to other GP-based methods, and it gives reliable estimations of the predictive uncertainties.

In our paper we propose to split the multi-fidelity surrogate modelling problem into two regression problems. The first problem is the single-fidelity regression for the low-fidelity code. The second problem is the regression for the high-fidelity code knowing the predictions and the predictive uncertainties of the low-fidelity surrogate model. GP regression allows prediction with quantified uncertainty for the low-fidelity code, which is important to minimise the predictive error and to quantify the predictive uncertainty of the surrogate model of the high-fidelity code, as we will see below. As in [15], we want to use a BNN for the regression knowing the low-fidelity prediction. The contribution we propose is an original strategy to transfer the low-fidelity predictive uncertainties to the BNN. For that we take well-chosen realisations of the predictor of the low-fidelity code by a quasi-Monte Carlo method based on Gauss-Hermite quadrature nodes, and we give them as inputs of the BNN in addition to the high-fidelity code inputs. A predictor is obtained by a weighted average of the BNN outputs corresponding to the different realisations. The predictive variance can also be assessed with the same sample.

The paper is organized as follows. Section 2 presents different methods to build single-fidelity surrogate models. The complete multi-fidelity method is presented in section 3. The specific interaction between GP regression and BNN is explained in section 4. Section 5 shows numerical results. Based on these results, the interest of the method is discussed in section 6.

2 Background: Regression with uncertainty quantification

In this section classical surrogate modelling methods with uncertainty quantification are presented. The GP regression method is presented in section 2.1. The BNN method is presented in section 2.2. Here we want to predict the scalar output $y = f(\mathbf{x})$, with $y \in I\!R$, of a computer code with input $\mathbf{x} \in I\!R^d$ from data set $(\mathbf{x}_i, y_i)_{i=1}^N$ with $y_i = f(\mathbf{x}_i)$.

2.1 Gaussian process regression

GP regression can be used to emulate a computer code with uncertainty quantification [26]. The output model as a function of the input $\mathbf{x}$ is a Gaussian process $Y(\mathbf{x})$ with mean $\mu(\mathbf{x})$ and stationary covariance function $C(\mathbf{x}, \mathbf{x}')$. Consequently, the posterior distribution of the output $Y(\mathbf{x})$ given $Y(\mathbf{x}_1) = y_1, \ldots, Y(\mathbf{x}_N) = y_N$ is Gaussian with mean:

$$\mu_\star(\mathbf{x}) = \mu(\mathbf{x}) + \mathbf{r}(\mathbf{x})^T \mathbf{C}^{-1}(\mathbf{y} - \boldsymbol{\mu}), \tag{1}$$

and covariance :

$$C_\star(\mathbf{x}, \mathbf{x}') = C(\mathbf{x}, \mathbf{x}') - \mathbf{r}(\mathbf{x})^T \mathbf{C}^{-1} \mathbf{r}(\mathbf{x}'), \tag{2}$$

with the vector $\mathbf{r}(\mathbf{x}) = (C(\mathbf{x}, \mathbf{x}_1), \ldots, C(\mathbf{x}, \mathbf{x}_N))^T$, the matrix $\mathbf{C}$ defined by $C_{i,j} = C(\mathbf{x}_i, \mathbf{x}_j)$, the vector $\boldsymbol{\mu} = (\mu(\mathbf{x}_1), \ldots, \mu(\mathbf{x}_N))^T$ and the vector $\mathbf{y} = (y_1, \ldots, y_N)^T$. The covariance function is chosen within a parametric family of kernels, whose parameters are fitted by maximizing the log marginal likelihood of the data, see [26, Chapter 2.2]. For practical applications the implementation of [26, Algorithm 2.1] can be used.

2.2 Bayesian neural network

Neural networks have been used to emulate unknown functions based on data [2], and in particular computer codes [25]. Our goal, however, is also to quantify the uncertainty of the emulation. BNN makes it possible to quantify predictive uncertainties. Below we present the BNN structure and the priors for the parameters.

We present a BNN with one hidden layer. Let N_l be the number of neurons in the hidden layer. The output of the first layer is

$$\mathbf{y}_1 = \Phi\left(\mathbf{w}_1 \mathbf{x} + \mathbf{b}_1\right), \tag{3}$$

with $\mathbf{x} \in I\!R^d$ the input vector of the BNN, $\mathbf{b}_1 \in I\!R^{N_l}$ the bias vector, $\mathbf{w}_1 \in I\!R^{N_l \times d}$ the weight matrix and $\mathbf{y}_1 \in I\!R^{N_l}$ the output of the hidden layer. The function

$\Phi : I\!R^{N_l} \to I\!R^{N_l}$ is of the form $\Phi(\mathbf{b}) = (\phi(b_j))_{j=1}^{N_l}$, where the activation function ϕ can be hyperbolic tangent or ReLU for instance. The second (and last) layer is fully linear:

$$BNN(\mathbf{x}) = \mathbf{w}_2^T \mathbf{y}_1 + \mathbf{b}_2, \tag{4}$$

with $\mathbf{w}_2 \in I\!R^{N_n}$ the weight matrix, $\mathbf{b}_2 \in I\!R$ the bias vector and $BNN(\mathbf{x}) \in I\!R$ the scalar output of the BNN at point $\mathbf{x}$.

We use a Bayesian framework similar to the one presented in [8]. Let $\boldsymbol{\theta}$ denote the parameter vector of the BNN, which is here $\boldsymbol{\theta} = (\mathbf{w}_i, \mathbf{b}_i)_{i=1,2}$. The probability distribution function (pdf) of the output given $\mathbf{x}$ and $\boldsymbol{\theta}$ is

$$p(y|\mathbf{x}, \boldsymbol{\theta}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(- \frac{(y - BNN_{\boldsymbol{\theta}}(\mathbf{x}))^2}{2\sigma^2}\Big), \tag{5}$$

where σ^2 is the variance of the random noise added to account for the fact that the neural network is an approximation. $BNN_{\boldsymbol{\theta}}(\mathbf{x})$ is the output of the neural network with parameter $\boldsymbol{\theta}$ at point $\mathbf{x}$.

Here we choose a prior distribution for $(\boldsymbol{\theta}, \sigma)$ that is classic in the field of BNN [8, Part 5]. The prior laws of the parameters $(\mathbf{w}_i, \mathbf{b}_i)_{i=1,2}$ are:

$$\mathbf{w}_i \sim \mathcal{N}\left(\mathbf{0}, \sigma_{w_i}^2 \mathbf{I}\right), \quad \mathbf{b}_i \sim \mathcal{N}\left(\mathbf{0}, \sigma_{b_i}^2 \mathbf{I}\right), \quad i = 1, 2, \tag{6}$$

with $\sigma_{w,b_{1,2}}$ the prior standard deviations. The prior for σ is the standard Gaussian $\mathcal{N}(0, 1)$ (assuming the function f has been normalized to be of order one). All parameters are assumed to be independent.

Applying Bayes' theorem, the posterior pdf of $(\boldsymbol{\theta}, \sigma)$ given the data $\mathcal{D} = (\mathbf{x}_i, y_i)_{i=1}^N$ is

$$p(\boldsymbol{\theta}, \sigma|\mathcal{D}) = \prod_{i=1}^{N} p(y_i|\mathbf{x}_i, \boldsymbol{\theta}, \sigma) p(\boldsymbol{\theta}, \sigma) \tag{7}$$

up to a multiplicative constant, where $p(\boldsymbol{\theta}, \sigma)$ is the prior distribution of $\boldsymbol{\theta}, \sigma$ described above. The posterior distribution of the output at $\mathbf{x}$ has pdf

$$p(y|\mathbf{x}, \mathcal{D}) = \iint p(y|\mathbf{x}, \boldsymbol{\theta}, \sigma) p(\boldsymbol{\theta}, \sigma|\mathcal{D}) d\boldsymbol{\theta} d\sigma, \tag{8}$$

and the two first moments are:

$$\mathbb{E}_{\text{post}}\left[Y\right] = \iint BNN_{\boldsymbol{\theta}}(\mathbf{x}) p(\boldsymbol{\theta}, \sigma|\mathcal{D}) d\boldsymbol{\theta} d\sigma, \tag{9}$$

$$\mathbb{E}_{\text{post}}\left[Y^2\right] = \iint \left(BNN_{\boldsymbol{\theta}}(\mathbf{x})^2 + \sigma^2\right) p(\boldsymbol{\theta}, \sigma|\mathcal{D}) d\boldsymbol{\theta} d\sigma. \tag{10}$$

Contrarily to GP regression, the prediction of a BNN cannot be expressed analytically as shown by (8) but there exist efficient sampling methods. To sample the posterior distribution of the BNN output, we need to sample the posterior distribution of $(\boldsymbol{\theta}, \sigma)$. In this paper the No-U-Turn Sampler (NUTS), which is a Hamiltonian Monte-Carlo (HMC) method, is used to sample the posterior distribution of $(\boldsymbol{\theta}, \sigma)$ [7]. By eqs. (9) and (10), the estimated mean $\tilde{f}$ and variance $\tilde{V}$ of the output at point $\mathbf{x}$ are:

$$\tilde{f}(\mathbf{x}) = \frac{1}{N_v} \sum_{i=1}^{N_v} BNN_{\boldsymbol{\theta}_i}(\mathbf{x}), \tag{11}$$

$$\tilde{V}(\mathbf{x}) = \frac{1}{N_v} \sum_{i=1}^{N_v} \left[BNN_{\boldsymbol{\theta}_i}(\mathbf{x}) - \tilde{f}(\mathbf{x}) \right]^2 + \frac{1}{N_v} \sum_{i=1}^{N_v} \sigma_i^2, \tag{12}$$

where the $(\boldsymbol{\theta}_i, \sigma_i)_{i=1}^{N_v}$ is the HMC sample of $(\boldsymbol{\theta}, \sigma)$ with its posterior distribution.

3 Combining GP regression and BNN

From now on we consider a multi-fidelity framework with two levels of code, high f_H and low f_L fidelity, as in [10]. The input is $\mathbf{x} \in I\!R^d$ and the outputs of both computer code levels $f_L(\mathbf{x})$ and $f_H(\mathbf{x})$ are scalar. We have access to N_L low-fidelity points and N_H high-fidelity points, with $N_H < N_L$. In our article we focus on the small data framework where the low-fidelity code is not perfectly known. Under such circumstances it remains uncertainty in the low-fidelity surrogate model. If $N_H \ll N_L$ the situation would be different and we could assume that the low-fidelity emulator is perfect.

We therefore have two surrogate modelling tools: GP regression and BNN. To do multi-fidelity with non-linear interactions the standard methods use combinations of regression methods. With our two methods we can make four combinations: GP-GP also called DeepGP in [19, 3], GP-BNN the method proposed in our paper, BNN-GP and BNN-BNN. The Deep GP model will be compared to the proposed method in all examples of our paper. The BNN-BNN method would be extremely expensive and very close to the full NN methods by adding the predictive uncertainty. The logic behind our choice of GP-BNN over BNN-GP is as follows: if we assume that the low-fidelity code is simpler than the high-fidelity code, then it must be approximated by a simpler model. GP regression is a surrogate model easier to obtain and it gives a Gaussian output distribution that can be sampled easily. Whereas BNN is more complex to construct and allows more varied output distributions to be emulated.

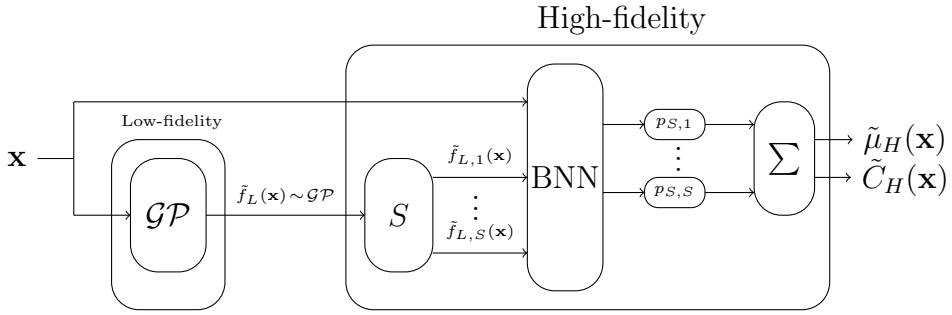


Figure 1: Schematic of the multi-fidelity model. The input is a point x. The output consists of a predictive mean $\tilde{\mu}_H(\mathbf{x})$ and a predictive variance $\tilde{C}_H(\mathbf{x})$

The code output to estimate is f_H with the help of low- and high-fidelity points. As the low-fidelity code f_L is not completely known a regression method with uncertainty quantification, GP regression, is used, to emulate it, as in section 2.1. The output of the low-fidelity GP is then integrated into the input to a BNN, described in section 2.2, to predict f_H.

The low-fidelity surrogate model is a GP built from N_L low-fidelity data points $(\mathbf{x}_{L,i}, f_L(\mathbf{x}_{L,i})) \in I\!R^d \times I\!R$. The optimisation of the hyper-parameters of the GP is carried out in the construction of the surrogate model. The GP is characterized by a predictive mean $\mu_L(\mathbf{x})$ and a predictive covariance $C_L(\mathbf{x}, \mathbf{x}')$.

To connect the GP with the BNN the simplest way is to concatenate $\mathbf{x}$ and $\mu_L(\mathbf{x})$ (the best low-fidelity predictor) as input to the BNN. However, this does not take into account the predictive uncertainty. Consequently, we may want to add $C_L(\mathbf{x}, \mathbf{x})$ or $\sqrt{C_L(\mathbf{x}, \mathbf{x})}$ to the input vector of the BNN. The idea is that the BNN could learn from the low-fidelity GP more than from its predictive mean only, in order to give reliable predictions of the high-fidelity code with quantified uncertainties.

We have investigated four methods to quantify uncertainty and to combine the two surrogate models.

It is possible to extend this method for more than two levels of codes. This is done by considering the output of the BNN as an input to the higher fidelity model. Two different models can therefore be considered. One uses the output of the BNN directly and by sampling gives it as input to the higher fidelity BNN. We can also generate realisations of our surrogate model and consider them as realisations of the low fidelity code and thus apply the GPBNN method.

4 Sampling methods

The two original learning sets are $\mathcal{D}^L = \{(\mathbf{x}_i^L, f_L(\mathbf{x}_i^L)), i = 1, \ldots, N_L\}$ and $\mathcal{D}^H = \{(\mathbf{x}_i^H, f_H(\mathbf{x}_i^H)), i = 1, \ldots, N_H\}$ with typically $N_H < N_L$ and we do not assume that the sets $\{x_i^L, i = 1, \ldots, N_L\}$ and $\{x_i^H, i = 1, \ldots, N_H\}$ are nested.

The low-fidelity model is emulated using GP regression, as a consequence the result is formulated as a posterior distribution given $\mathcal{D}^L$ that has the form of a Gaussian law.

Proposition 1. *The posterior distribution of $Y_L(\mathbf{x})$ knowing $\mathcal{D}^L$ is the Gaussian distribution with mean $\mu_L(\mathbf{x})$ and variance $\sigma_L^2(\mathbf{x})$ of the form (1-2). We denote its pdf by $p(y_L|\mathcal{D}_L, \mathbf{x})$.*

Proof. The proof is given in [26, chapter 2.2] (prediction with noise free observations). □

The posterior distribution of the high-fidelity code given the low-fidelity learning set $\mathcal{D}^L$ and the high-fidelity learning set $\mathcal{D}^H$ may have different forms depending on the input of the BNN.

4.1 Mean-Standard deviation method

In the Mean-Standard deviation method, called Mean-Std method, we give as input to the BNN the point $\mathbf{x}$ and the information usually available at the output of a GP regression, i.e. the predictive mean and standard deviation of the low-fidelity emulator at the point $\mathbf{x}$.

In this method, the input of the BNN whose output gives the prediction of the high-fidelity code at $\mathbf{x}$ is $(\mathbf{x}, \mu_L, \sigma_L)$. The idea is that the BNN input consists of the input $\mathbf{x}$ of the code and of the mean and standard deviation of the posterior distribution of the low-fidelity emulator at $\mathbf{x}$. We use the learning set $\mathcal{D}_{MS}^H = \big\{\big(\mathbf{x}_i^H, \mu_L(\mathbf{x}_i^H), \sigma_L(\mathbf{x}_i^H), f_H(\mathbf{x}_i^H)\big), i = 1, \ldots, N_H\big\}$ to train the BNN. Note that $\mathcal{D}_{MS}^H$ can be deduced from $\mathcal{D}^L$ and $\mathcal{D}^H$.

Proposition 2. *The posterior distribution of $Y_H(\mathbf{x})$ knowing $\mathcal{D}^L$ and $\mathcal{D}_{MS}^H$ has pdf*

$$p\big(y_H|\mathbf{x}, \mathcal{D}_{MS}^H, \mathcal{D}^L\big) = \iint \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(-\frac{(y_H - BNN_{\boldsymbol{\theta}}(\mathbf{x}, \mu_L(\mathbf{x}), \sigma_L(\mathbf{x})))^2}{2\sigma^2}\Big) p(\boldsymbol{\theta}, \sigma|\mathcal{D}_{MS}^H) d\sigma d\boldsymbol{\theta}, \tag{13}$$

with $p(\boldsymbol{\theta}, \sigma|\mathcal{D}_{MS}^H)$ the posterior pdf of the hyper-parameters of the BNN.

Corollary 3. *The mean and variance of the posterior distribution of* $Y_H(\mathbf{x})$ *knowing* $\mathcal{D}^H_{MS}$ *and* $\mathcal{D}^L$ *is:*

$$\mu_H(\mathbf{x}) = \iint BNN_{\boldsymbol{\theta}}(\mathbf{x}, \mu_L(\mathbf{x}), \sigma_L(\mathbf{x}))p(\boldsymbol{\theta}, \sigma|\mathcal{D}^H)d\sigma d\boldsymbol{\theta}, \tag{14}$$

and

$$C_H(\mathbf{x}) = \iint \left(BNN_{\boldsymbol{\theta}}(\mathbf{x}, \mu_L(\mathbf{x}), \sigma_L(\mathbf{x}))^2 + \sigma^2\right) p(\boldsymbol{\theta}, \sigma|\mathcal{D}^H)d\sigma d\boldsymbol{\theta}, \tag{15}$$

Proposition 4. *With* $\mathcal{D}^H_{MS}$*, the estimators* $\tilde{\mu}_H(\mathbf{x})$*:*

$$\tilde{f}_H(\mathbf{x}) = \frac{1}{N_v}\sum_{i=1}^{N_v} BNN_{\boldsymbol{\theta}_i}(\mathbf{x}, \tilde{\mu}_L(\mathbf{x}), \tilde{\sigma}_L(\mathbf{x})), \tag{16}$$

and

$$\tilde{C}_H(\mathbf{x}) = \frac{1}{N_v}\sum_{i=1}^{N_v} \left(BNN_{\boldsymbol{\theta}_i}(\mathbf{x}, \tilde{\mu}_L(\mathbf{x}), \tilde{\sigma}_L(\mathbf{x}))^2 + \sigma_i^2\right) - \tilde{\mu}_H(\mathbf{x})^2. \tag{17}$$

The low fidelity data and model are described in proposition 2.

The estimators needed samples of the posterior distribution of $\boldsymbol{\theta}$ and σ to predict mean and variance. By the HMC method (NUTS), we get the posterior law of $(\boldsymbol{\theta}_j, \sigma_j)$ the hyperparameters of the high-fidelity model.

4.2 Quantiles method

The Quantiles method consists of giving the mean and two quantiles of the low-fidelity GP emulator as input to the BNN. Assuming we want to have the high-fidelity output uncertainty at level $\alpha\%$ we take the $\alpha/2\%$ and the $(1-\alpha/2)\%$ quantiles. The expression of the BNN input vector is $\mathbf{x}^{\text{BNN}} = (\mathbf{x}_i^H, \mu_L(\mathbf{x}_i^L), Q_{L,\alpha}(\mathbf{x}_i^H), Q_{L,(1-\alpha)}(\mathbf{x}_i^H))$ then the high-fidelity learning set is $\mathcal{D}^H_Q : \left\{\left((\mathbf{x}_i^H, \mu_L(\mathbf{x}_i^L), Q_{L,\alpha}(\mathbf{x}_i^H), Q_{L,(1-\alpha)}(\mathbf{x}_i^H)), f_H(\mathbf{x}_i^H)\right), i = 1, \cdots, N_L\right\}$.

Proposition 5. *The posterior distribution of* $Y_H(\mathbf{x})$ *knowing* $\mathcal{D}^L$ *and* $\mathcal{D}^H_Q$ *has pdf*

$$p\big(y_H|\mathbf{x}, \mathcal{D}^H_Q, \mathcal{D}^L\big) = \iint \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(-\frac{(y_H - BNN_{\boldsymbol{\theta}}(\mathbf{x}, \mu_L(\mathbf{x}), Q_{L,\alpha}(\mathbf{x}_i^H), Q_{L,(1-\alpha)}(\mathbf{x}_i^H)))^2}{2\sigma^2}\Big)p(\boldsymbol{\theta}, \sigma|\mathcal{D}^H_Q)d\sigma d\boldsymbol{\theta}, \tag{18}$$

with $p(\boldsymbol{\theta}, \sigma|\mathcal{D}^H_{MS})$ *the posterior pdf of the hyper-parameters of the BNN.*

Corollary 6. *The mean and variance of the posterior distribution of $Y_H(\mathbf{x})$ knowing $\mathcal{D}^H_{MS}$ and $\mathcal{D}^L$ is:*

$$\mu_H(\mathbf{x}) = \iint BNN_{\boldsymbol{\theta}}(\mathbf{x}, \mu_L(\mathbf{x}), \tilde{Q}_{L,\alpha}(\mathbf{x}_i^H), \tilde{Q}_{L,(1-\alpha)}(\mathbf{x}_i^H)) p(\boldsymbol{\theta}, \sigma | \mathcal{D}^H_Q) d\sigma d\boldsymbol{\theta}, \quad (19)$$

and

$$C_H(\mathbf{x}) = \iint \left(BNN_{\boldsymbol{\theta}}(\mathbf{x}, \mu_L(\mathbf{x}), \tilde{Q}_{L,\alpha}(\mathbf{x}_i^H), \tilde{Q}_{L,(1-\alpha)}(\mathbf{x}_i^H))^2 + \sigma^2 \right) p(\boldsymbol{\theta}, \sigma | \mathcal{D}^H_Q) d\sigma d\boldsymbol{\theta}, (20)$$

Proposition 7. *The estimators $\tilde{\mu}_H(\mathbf{x})$, $\tilde{C}_H(\mathbf{x})$ are estimators of the mean and variance expression in corollary 6 with $\mathcal{D}^H_Q$ the α quantiles of the GP posterior law. The mean and variance are:*

$$\tilde{\mu}_H(\mathbf{x}) = \frac{1}{N_v} \sum_{i=1}^{N_v} BNN_{\boldsymbol{\theta}_i}(\mathbf{x}, \tilde{\mu}_L(\mathbf{x}), \tilde{Q}_{L,\alpha}(\mathbf{x}), \tilde{Q}_{L,(1-\alpha)}(\mathbf{x})), \quad (21)$$

$$\tilde{C}_H(\mathbf{x}) = \frac{1}{N_v} \sum_{i=1}^{N_v} \left(BNN_{\boldsymbol{\theta}_i}(\mathbf{x}, \tilde{\mu}_L(\mathbf{x}), \tilde{Q}_{L,\alpha}(\mathbf{x}), \tilde{Q}_{L,(1-\alpha)}(\mathbf{x}))^2 + \sigma_i^2 \right) - \tilde{\mu}_H(\mathbf{x})^2. \quad (22)$$

This method is very similar to the Mean-Std method and only the BNN entries change. This is why the estimators have the same form. It can be noted that the dimension of the BNN inputs is larger than for the Mean-Std method. The estimators needed samples of the posterior distribution of $\boldsymbol{\theta}$ and σ to predict mean and variance. By the HMC method (NUTS), we get the posterior law of $(\boldsymbol{\theta}_j, \sigma_j)$ the hyperparameters of the high-fidelity model.

4.3 Random Samples method

The Random sample method method is based on Monte Carlo method. We generate S realisations of the posterior distribution of the low-fidelity surrogate model. To get the BNN output mean and variance we compute the empirical mean and variance with respect to the S realisations.

The GP posterior distribution is one-dimensional and Gaussian for each value of $\mathbf{x}$. Therefore, we can sample the low-fidelity posterior law of the GP. The samples are called $\tilde{f}_{L,j}(\mathbf{x}_i^H)$ for $j = 1, \cdots, S$. In order to get samples from $\mu_L(\mathbf{x})$ and $\sigma_L(\mathbf{x})$ the formula is:

$$\tilde{f}_{L,j}(\mathbf{x}_i^H) = \mu_L(\mathbf{x}_i^H) + \sigma_L(\mathbf{x}_i^H)\epsilon_j, \quad (23)$$

where ϵ_j for $j = 1, \cdots, S$ are reduced centred Gaussian iid samples. The input vector of the BNN is $\mathbf{x}^{\text{BNN}} = (\mathbf{x}_i^H, \tilde{f}_{L,j}(\mathbf{x}_i^H))$. The high-fidelity learning set becomes $\mathcal{D}^H_{RS} : \left\{ \left((\mathbf{x}_i^H, \tilde{f}_{L,j}(\mathbf{x}_i^H)), f_H(\mathbf{x}_i^H) \right), i = 1, \cdots, N_L, j = 1, \cdots, S \right\}$.

Proposition 8. *The posterior distribution of* $Y_H(\mathbf{x})$ *knowing* $\mathcal{D}^H_{RS}$ *and* $\mathcal{N}\left(\mu_L(\mathbf{x}), \sigma^2_L(\mathbf{x})\right)$ *is*

$$p\left(y_H|\mathbf{x}, \mathcal{D}^H_{RS}, \mathcal{D}^L\right) = \iiint \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(y_H - BNN_{\boldsymbol{\theta}}(\mathbf{x}, y_L))^2\right) p(\boldsymbol{\theta}, \sigma|\mathcal{D}^H_{RS}) p(y_L|\mathbf{x}, \mathcal{D}^L) d\sigma d\boldsymbol{\theta} dy_L, \tag{24}$$

with $p(\boldsymbol{\theta}, \sigma|\mathcal{D}^H)$ *the posterior distribution of the hyper-parameters of the BNN and* $\mathbf{x}^{BNN}$ *the input vector of the BNN.*

Corollary 9. *The posterior mean of* $Y_H(\mathbf{x})$ *is:*

$$\mu_H(\mathbf{x}) = \iiint BNN_{\boldsymbol{\theta}}(\mathbf{x}, y_L) p(\boldsymbol{\theta}, \sigma|\mathcal{D}^H_{RS}) p(y_L|\mathbf{x}, \mathcal{D}^L) d\sigma d\boldsymbol{\theta} dy_L. \tag{25}$$

The posterior variance of $Y_H(\mathbf{x})$ *is:*

$$C_H(\mathbf{x}) = \iiint \left(BNN^2_{\boldsymbol{\theta}}(\mathbf{x}, y_L) + \sigma^2\right) p(\boldsymbol{\theta}, \sigma|\mathcal{D}^H_{RS}) p(y_L|\mathbf{x}, \mathcal{D}^L) d\sigma d\boldsymbol{\theta} dy_L - \mu^2_H(\mathbf{x}). \tag{26}$$

Corollary 10. *The estimators* $\tilde{\mu}_H(\mathbf{x})$, $\tilde{C}_H(\mathbf{x})$ *of* $\mu_H(\mathbf{x})$, $C_H(\mathbf{x})$ *are defined by:*

$$\tilde{\mu}_H(\mathbf{x}) = \frac{1}{N_v S} \sum_{i=1}^{N_v} \sum_{j=1}^{S} BNN_{\boldsymbol{\theta}_i}(\mathbf{x}, \tilde{f}_{L,j}(\mathbf{x})), \tag{27}$$

$$\tilde{C}_H(\mathbf{x}) = \frac{1}{N_v S} \sum_{i=1}^{N_v} \sum_{j=1}^{S} BNN_{\boldsymbol{\theta}_i}(\mathbf{x}, \tilde{f}_{L,j}(\mathbf{x}))^2 + \frac{1}{N_v} \sum_{i=1}^{N_v} \sigma_i^2 - \tilde{\mu}_H(\mathbf{x})^2, \tag{28}$$

where $\tilde{f}_{j,L}(\mathbf{x})$ *for* $j = 1, \cdots, S$ *are iid with the distribution given in proposition 1 and* $\boldsymbol{\theta}_i$ *for* $i = 1, \cdots, N_v$ *are samples of the posterior pdf* $p(\boldsymbol{\theta}, \sigma|\mathcal{D}^H_{RS})$ *obtained by using MCMC algorithm.*

The estimators needed samples of the posterior distribution of $\boldsymbol{\theta}$ and σ to predict mean and variance. By the HMC method (NUTS), we get the posterior law of $(\boldsymbol{\theta}_j, \sigma_j)$ the hyperparameters of the high-fidelity model.

Proof. We have samples $\tilde{f}_{L,j}(\mathbf{x}))^2$ of the low-fidelity distribution. Using Monte-Carlo method we have:

$$\int BNN_{\boldsymbol{\theta}}(\mathbf{x}^{\mathrm{BNN}}) p(y_L(\mathbf{x})|\mathcal{D}^L) dy_L \approx \frac{1}{S} \sum_{j=1}^{S} BNN_{\boldsymbol{\theta}}(\mathbf{x}^{\mathrm{BNN}}) \tag{29}$$

The sampling $\boldsymbol{\theta}_i$ on the parameters $\boldsymbol{\theta}$ gives us that:

$$\iint BNN_{\boldsymbol{\theta}}(\mathbf{x}^{\mathrm{BNN}})p(\boldsymbol{\theta}|\mathcal{D}^H)p(y_L(\mathbf{x})|\mathcal{D}^L)d\boldsymbol{\theta}dy_L \approx \frac{1}{N_v S}\sum_{i=1}^{N_v}\sum_{j=1}^{S} BNN_{\boldsymbol{\theta}_i}(\mathbf{x}, \tilde{f}_{L,j}(\mathbf{x}))(30)$$

For the variance we have the same term but with square. The estimation is then:

$$\tilde{C}_H(\mathbf{x}) = \frac{1}{N_v}\sum_{i=1}^{N_v}\sum_{j=1}^{S}\left(\frac{1}{S}BNN_{\boldsymbol{\theta}_i}(\mathbf{x}, \tilde{f}_{L,j}(\mathbf{x}))^2 + \frac{1}{S}\sigma_i^2\right) - \tilde{\mu}_H(\mathbf{x})^2, \tag{31}$$

with $\sum_{j=1}^{S}\frac{1}{S} = 1$ we have an estimator of the variance. □

4.4 Gauss-Hermite quadrature

The Gauss-Hermite method is based on a quasi Monte Carlo method and is presented in fig. 1. For a given point $\mathbf{x}$, the point $\mathbf{x}$ and several well chosen realisations $(\tilde{f}_{L,i}(\mathbf{x}))_{i=1,\dots,S}$ of the GP at this point are given as input to the BNN. We then obtain realisations of the BNN which we combine to obtain the output distribution of the prediction of the high-fidelity code.

The GP posterior distribution is one-dimensional and Gaussian for each value of $\mathbf{x}$. Therefore, a deterministic method for sampling is preferable in order to limit the number of calls to the BNN. In the following this method is called GPBNN. We propose to sample the Gaussian distribution by a quasi Monte Carlo method using S Gauss-Hermite quadrature nodes [5, Chapter 3]. This method has been chosen because it gives the best interpolation of a Gaussian distribution. The samples $\tilde{f}_{L,i}(\mathbf{x})$, with $i = 1, \dots, S$, of the GP posterior distribution are constructed using the roots $z_{S,i}$ of the physicists' version of the Hermite polynomials $H_S(x) = (-1)^S e^{x^2}\partial_x^S e^{-x^2}$, $S \in \mathbb{N}$. For each input $\mathbf{x}$ the GP posterior law has mean $\mu_L(\mathbf{x})$ and variance $C_L(\mathbf{x}, \mathbf{x})$. Therefore, the ith realisation in the Gauss-Hermite quadrature formula is:

$$\tilde{f}_{L,i}(\mathbf{x}) = \mu_L(\mathbf{x}) + z_{S,i}\sqrt{2}\sqrt{C_L(\mathbf{x}, \mathbf{x})}, \tag{32}$$

and the associated weight is $p_{S,i} = \frac{2^{S-1}S!}{S^2 H_{S-1}^2(z_{S,i})}$, for $i = 1, \dots, S$. The BNN has as a learning set of $\mathcal{D}_{GH}^H$: $\left\{(\mathbf{x}_i^H, \tilde{f}_{L,j}(\mathbf{x}_i^H), i = 1, \cdots, N_L, j = 1, \cdots, S\right\}$. Its output is called $BNN_{\boldsymbol{\theta}}(\mathbf{x}, \tilde{f}_{L,i}(\mathbf{x}))$. In order to obtain the posterior distribution of $\tilde{f}_H$ knowing the input $\mathbf{x}$, $(\boldsymbol{\theta}, \sigma)$ is sampled by the HMC method (NUTS). We get N_v samples $(\boldsymbol{\theta}_j, \sigma_j)$ with $j = 1, \dots, N_v$.

Proposition 11. *The posterior distribution of* $Y_H(\mathbf{x})$ *knowing* $\mathcal{D}_{GH}^H$ *and* $\mathcal{N}\left(\mu_L(\mathbf{x}), \sigma_L^2(\mathbf{x})\right)$ *is*

$$p\left(y_H|\mathbf{x}, \mathcal{D}_{GH}^H, \mathcal{D}^L\right) = \iiint \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(y_H - BNN_{\boldsymbol{\theta}}(\mathbf{x}, y_L))^2\right) p(\boldsymbol{\theta}, \sigma|\mathcal{D}_{RS}^H) p(y_L|\mathbf{x}, \mathcal{D}^L) d\sigma d\boldsymbol{\theta} dy_L, \tag{33}$$

with $p(\boldsymbol{\theta}, \sigma|\mathcal{D}_{GH}^H)$ *the posterior distribution of the hyper-parameters of the BNN and* $\mathbf{x}^{BNN}$ *the input vector of the BNN.*

Corollary 12. *The posterior mean of* $Y_H(\mathbf{x})$ *is:*

$$\mu_H(\mathbf{x}) = \iiint BNN_{\boldsymbol{\theta}}(\mathbf{x}, y_L) p(\boldsymbol{\theta}, \sigma|\mathcal{D}_{GH}^H) p(y_L|\mathbf{x}, \mathcal{D}^L) d\sigma d\boldsymbol{\theta} dy_L. \tag{34}$$

The posterior variance of $Y_H(\mathbf{x})$ *is:*

$$C_H(\mathbf{x}) = \iiint \left(BNN_{\boldsymbol{\theta}}^2(\mathbf{x}, y_L) + \sigma^2\right) p(\boldsymbol{\theta}, \sigma|\mathcal{D}_{GH}^H) p(y_L|\mathbf{x}, \mathcal{D}^L) d\sigma d\boldsymbol{\theta} dy_L - \mu_H^2(\mathbf{x}). \tag{35}$$

Corollary 13. *We use the Gauss-Hermite quadrature of the low-fidelity GP, the high-fidelity learning set is* $\mathcal{D}_{GH}^H$*, with* $\tilde{f}_{L,i}(\mathbf{x})$ *given at eq.* (32)*. The estimator of the high-fidelity mean of the output of the high-fidelity model is:*

$$\tilde{\mu}_H(\mathbf{x}) = \frac{1}{N_v} \sum_{i=1}^{N_v} \sum_{j=1}^{S} p_{S,j} BNN_{\boldsymbol{\theta}_i}(\mathbf{x}, \tilde{f}_{L,j}(\mathbf{x})), \tag{36}$$

and by corollary 12 the estimator of the variance is:

$$\tilde{C}_H(\mathbf{x}) = \frac{1}{N_v} \sum_{i=1}^{N_v} \left(\sum_{j=1}^{S} p_{S,j} BNN_{\boldsymbol{\theta}_i}(\mathbf{x}, \tilde{f}_{L,j}(\mathbf{x})) \right)^2 + \frac{1}{N_v} \sum_{i=1}^{N_v} \sigma_i^2 - \tilde{\mu}_H^2(\mathbf{x}). \tag{37}$$

When $S \to +\infty$, $\sum_{j=1}^{N_v} \sum_{i=1}^{S} p_{S,i} BNN_{\boldsymbol{\theta}_j}(\mathbf{x}, \tilde{f}_{L,i}(\mathbf{x}))$ tends to the output of the BNN knowing the posterior law of the GP with parameters $\boldsymbol{\theta}_j$. Then when $N_v \to +\infty$, $\tilde{f}_H(\mathbf{x})$ tends to the posterior of the BNN. $\tilde{f}_H(\mathbf{x})$, given in eq. (36) tends to the mean of the posterior law of the high-fidelity when N_v and S tend to $+\infty$.

Note that this sampling method is different from the Quantiles method (even for $S = 3$ and $\alpha \approx 0.110$). Indeed, the Quantiles method gives us complete information on the law whereas the Gauss-Hermite method is a sampling method. The weights are different between the random samples and the Gauss-Hermite method.

The choice of S is a trade-off between a large value that is computationally costly and a small value that does not propagate the uncertainty appropriately.

$S = 2$ is the smallest admissible value regarding the information that should be transferred. At first glance, a large value of S could be expected to be the best choice in the point of view of the predictive accuracy. However, a too large value of S degrades the accuracy of the predictive mean estimation. This is due to large variations of $\tilde{f}_{L,i}(\mathbf{x})$ for large values of S. Interesting values turn out to be between 3 and 10 depending on the quality of the low-fidelity emulator, as discussed in section 5.

For N_v we try to take the highest possible value. However, the calculation time imposes a limit. We try to have the smallest value which in our case always converges.

We could expect the computational cost of the GPBNN method to be expensive. This is due to the fact that we combine Monte-Carlo methods. The number of operations needed to compute an iteration of the HMC optimisation is proportional to $S \times N_v \times N_H$. Because S and N_H are small in our context the computational cost of one realisation of the BNN is actually low. Thus optimisation of the hyperparameters seems feasible for $N_H \lesssim 100$.

5 Experiments

In this section we present two analytic examples and a simulated one. The first one is a 1D function, and we consider that the low-fidelity data may be unknown in a certain subdomain. The second one is a 2D function with noise. Finally, we test the strategy on a pendulum system. All the numerical experiments are carried out on a laptop (of 2017, dell precision with intel core i7) using only CPU and the running time never exceeds 2 hours.

5.1 1D function approximation

The low- and high-fidelity functions are:

$$f_L(x) = \sin 8\pi x, \qquad f_H(x) = \big(x - \sqrt{2}\big) f_L^2(x), \tag{38}$$

with $x \in [0, 1]$, where f_H is the high-fidelity function (code) and f_L the low-fidelity function (code). These functions have been introduced in [19] and are well estimated with a DeepGP and a quadratic form of the covariance. In this example we assume that we have access to a lot of low-fidelity data, $N_L = 100$, while the high-fidelity data is small, $N_H = 20$. We also consider situations in which there is a segment $\bar{I} \subset [0, 1]$ where we do not have access to $f_L(x)$. The learning set for the high-fidelity code is obtained by partitioning $[0, 1]$ into N_H segments with equal lengths and then by choosing independently one point randomly on each segment

with uniform distribution. The learning set for the low-fidelity code is obtained by partitioning $[0,1]\backslash\bar{I}$ into N_L segments with equal length and then by choosing independently one point randomly on each segment with uniform distribution. The test set is composed of $N_{\mathrm{T}} = 1000$ independent points following a random uniform law on $[0,1]$.

We denote by $\left(\mathbf{x}_{\mathrm{T}}^{(i)}, f_H(\mathbf{x}_{\mathrm{T}}^{(i)})\right)_{i=1,\dots,N_{\mathrm{T}}}$ the test set. The error is evaluated by:

$$Q_{\mathrm{T}}^2 = 1 - \frac{\sum_{i=1}^{N_{\mathrm{T}}} \left[\tilde{\mu}_H(\mathbf{x}_{\mathrm{T}}^{(i)}) - f_H(\mathbf{x}_{\mathrm{T}}^{(i)})\right]^2}{N_{\mathrm{T}} \mathbb{V}_{\mathrm{T}}\left(f_H\right)}, \tag{39}$$

with $\mathbb{V}_{\mathrm{T}}(f_H) = \frac{1}{N_{\mathrm{T}}} \sum_{i=1}^{N_{\mathrm{T}}} \left[f_H(\mathbf{x}_{\mathrm{T}}^{(i)}) - \frac{1}{N_{\mathrm{T}}} \sum_{j=1}^{N_{\mathrm{T}}} f_H(\mathbf{x}_{\mathrm{T}}^{(j)})\right]^2$. A highly predictive model gives a Q_{T}^2 close to 1 while a less predictive model has a smaller Q_{T}^2. The coverage probability CP_α is defined as the probability for the actual value of the function to be within the prediction interval with confidence level α of the surrogate model.

$$\mathrm{CP}_\alpha = \frac{1}{N_{\mathrm{T}}} \sum_{i=1}^{N_{\mathrm{T}}} \mathbf{1}_{f_H(\mathbf{x}_{\mathrm{T}}^{(i)}) \in \mathcal{I}_\alpha(\mathbf{x}_{\mathrm{T}}^{(i)})}, \tag{40}$$

with $\mathbf{1}$ the indicator function. The mean predictive interval width MPIW_α is the average width of the prediction intervals:

$$\mathrm{MPIW}_\alpha = \frac{1}{N_{\mathrm{T}}} \sum_{i=1}^{N_{\mathrm{T}}} \left|\mathcal{I}_\alpha(\mathbf{x}_{\mathrm{T}}^{(i)})\right|, \tag{41}$$

with $\mathcal{I}_\alpha(\mathbf{x})$ the prediction interval at point $\mathbf{x}$ with confidence level α and $|\mathcal{I}_\alpha(\mathbf{x})|$ the length of this interval. For the GPBNN method we obtain the interval $\mathcal{I}_\alpha(\mathbf{x})$ by sampling N_v realisations of the noisy BNN: $y_j = BNN_{\boldsymbol{\theta}_j}(\mathbf{x}) + \sigma_j \epsilon_j$ where $(\boldsymbol{\theta}_j, \sigma_j)_{j=1}^{N_v}$ is the HMC sample of the posterior distribution of $(\boldsymbol{\theta}, \sigma)$ and the ϵ_j's are iid Gaussian random variables with mean zero and variance 1. The interval $\mathcal{I}_\alpha(\mathbf{x})$ is the smallest interval that contains the fraction α of the realisations $(y_j)_{j=1}^{N_v}$. For the GP 1F model and the AR(1) model the prediction interval is centered at the predictive mean and its half-length is $q_{1-\frac{\alpha}{2}}$ times the predictive standard deviation, where $q_{1-\frac{\alpha}{2}}$ is the $1-\frac{\alpha}{2}$-quantile of the standard Gaussian law, because the posterior distributions are Gaussian. For the Deep GP model the prediction interval is obtained by Monte-Carlo sampling of the posterior distribution (with 1000 samples). For a fixed α, we want the CP_α to be as close to α as possible and MPIW_α to be as small as possible to have a good uncertainty quantification.

We use GP regression with zero mean function and tensorized Matérn 5/2 covariance function for the low-fidelity GP regression. The implementation we

Table 1: Error Q_T^2, coverage probability CP_α and mean predictive interval width MPIW_α for $\alpha = 80\%$ and for different methods of sampling. Here $\bar{I} = \emptyset$.

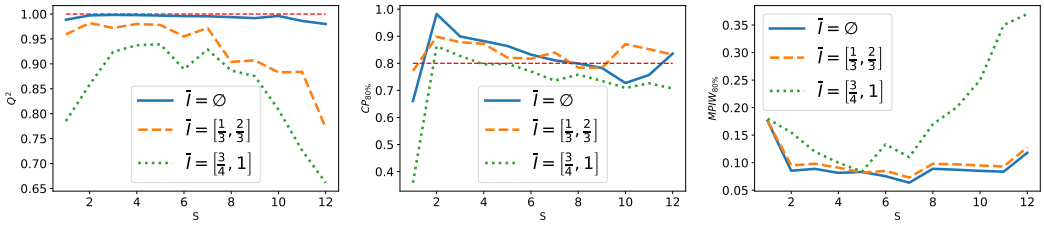
	Q_T^2	CP_α	MPIW_α
Gauss-Hermite $S = 5$	**0.99**	**0.88**	**0.083**
Mean-Std	**0.99**	0.97	0.095
Quantiles	**0.99**	0.90	0.105
Random Samples $S = 5$	**0.99**	0.98	0.92
Random Samples $S = 15$	**0.99**	0.96	0.090

use is from [6]. The optimisation for GP regression gives a correlation length of 0.108. For this example we choose $N_n = 30$ neurons, we use the ReLU function as activation function, and we use the BNN implementation proposed in [1]. The sample size of the posterior distribution of the BNN parameter $(\boldsymbol{\theta}, \sigma)$ is $N_v = 500$.

Low-fidelity surrogate models of different accuracies are considered to understand how our strategy behaves under low-fidelity uncertainty. This is done by considering that low-fidelity data points are only accessible in $[0, 1]\backslash\bar{I}$. We have thus chosen to study three cases, a very good low-fidelity emulator with $\bar{I} = \emptyset$ (for which the $Q^2_{l\to l}$ of the low-fidelity emulator is 0.99), a good emulator with $\bar{I} = [\frac{1}{3}, \frac{2}{3}]$ ($Q^2_{l\to l} = 0.98$) and a poor emulator with $\bar{I} = [\frac{3}{4}, 1]$ ($Q^2_{l\to l} = 0.84$).

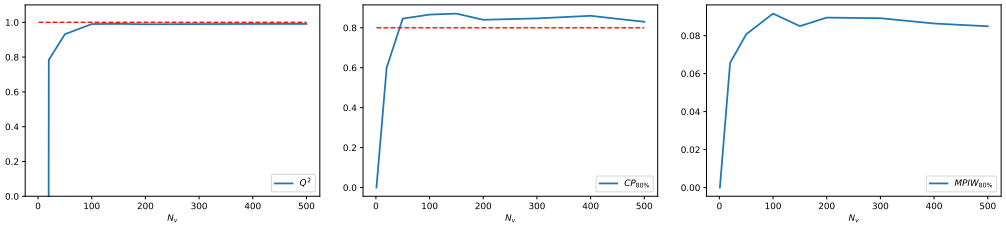
Table 1 compars for these examples the different sampling techniques, proposed in section 3, for $\bar{I} = \emptyset$. All methods have the same efficiency in terms of Q^2_{T}. The uncertainties of the predictions are overestimated for all methods. However, the Gauss-Hermite method has the best CP_α and the best uncertainty interval (i.e., the smallest mean predictive interval width MPIW_α). The quantiles method and the Mean-Std method also have reasonable CP_α, but their uncertainty intervals are larger. This leads us to use the Gauss-Hermite method. However, we note that all methods over-estimated the prediction interval. We believe this is due to the high regularity of the function to be predicted.

In fig. 2 we report the performances of the GPBNN method as functions of S between 1 and 12 for different $\bar{I}$. For $S = 1$ the uncertainty is underestimated and the accuracy of the prediction is not optimal, which shows that it is important to exploit the uncertainty predicted by the low-fidelity model. For $2 \leq S \leq 5$ the prediction is good, the error is constant and the Q^2_{T} is maximal as seen in fig. 2(a). Figure 2(b) shows that the coverage probability is acceptable for $2 \leq S \leq 7$.



(a) Q_T^2 (b) $\text{CP}_{80\%}$ (c) $\text{MPIW}_{80\%}$

Figure 2: Error Q_{T}^2, coverage probability at 80% and $\text{MPIW}_{80\%}$ as functions of S.



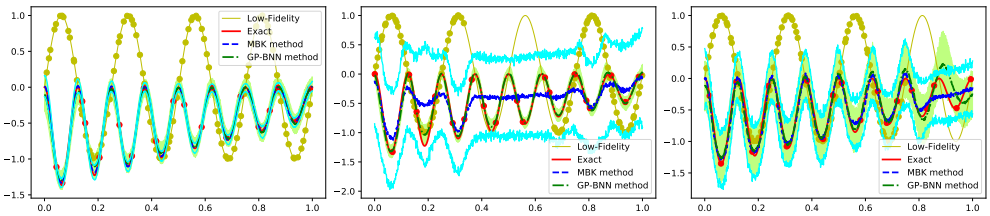
(a) Q_T^2 (b) $\text{CP}_{80\%}$ (c) $\text{MPIW}_{80\%}$

Figure 3: Error Q_{T}^2, coverage probability at 80% and $\text{MPIW}_{80\%}$ as functions of N_v.

Finally, the $\text{MPIW}_{80\%}$ on fig. 2(c) is minimal for $3 \leq S \leq 7$. The best value of S is in $[2, 5]$ depending on the accuracy of the low-fidelity emulator.

We have carried out a study on the best value of N_v. We thought that the best value would be the largest possible. Therefore, we tested for values of N_v ranging from 1 to 1000 for $\bar{I} = \emptyset$. For values of N_v greater than 200 the performance is identical as a function of N_v. For values below 200 a greater variability was found. We chose to use $N_v = 500$ to have a sufficient margin. In Figure 3 we have averaged the estimators for 5 independent training sets.

We now want to compare our GPBNN method with other ones. The single-fidelity GP method used to emulate the high-fidelity code from the N_H high-fidelity points is called GP 1F. We use the implementation in [6]. The multi-fidelity GP regression with autoregressive form introduced by [10] and improved by [12] is called autoregressive model AR(1). The method proposed in both [3, 19] is called DeepGP, we use the implementation from [19] and the covariance given in [3] equation (11). The method from [15] is called MBK method. The MBK method is the combination of a fully connected NN for low-fidelity regression and BNN for



(a) $\bar{I} = \emptyset$ (b) $\bar{I} = [\frac{1}{3}, \frac{2}{3}]$ (c) $\bar{I} = [\frac{3}{4}, 1]$

Figure 4: Comparison between the MBK method (in blue) and the GPBNN method with $S = 5$ (in green) for the estimation of the function f_H eq. (38) (in red). The high-fidelity data points are in red. The light colored lines (blue and green) plot the predictive intervals. The uncertainty interval is given by the $\mathcal{I}_{80\%}(\mathbf{x})$

high-fidelity. We implemented it using [1]. The methods that require the minimal assumptions on the function f_L and f_H are the GPBNN and MBK methods. This is the reason why they are the two methods that are compared in fig. 4.

First, the output laws for the MBK method and for the GPBNN presented in fig. 4 are different. For the MBK method the posterior law is associated to the high-fidelity BNN knowing both the high-fidelity data and the low-fidelity model. Unlike the MBK method, the GBBNN method's output law represents the posterior law knowing high-fidelity points and the posterior distribution of the low-fidelity model built from the N_L low-fidelity points. In fig. 4 $S = 5$ because, as discussed above, this value seems to be the best value when the low-fidelity code is not so accurate.

The presented techniques for multi-fidelity regression are compared in tables 2 and 3. The GP 1F model and the multi-fidelity AR(1) model do not have good predictive properties. For the GP 1F model it is due to the lack of high-fidelity data and for the AR(1) model it is due to the strongly nonlinear relationship between the code levels. The three other methods perform almost perfectly when $\bar{I} = \emptyset$. The DeepGP is outstanding when $\bar{I} = \emptyset$ but the interaction between the codes (a quadratic form) is given exactly in the covariance structure which is a strong assumption in DeepGP that is hard to verify in practical applications. When $\bar{I}$ is non-empty the MBK method gives less accurate predictions because the method assumes strong knowledge of the low-fidelity code level. However, its uncertainty interval seems realistic for this example although too large. The DeepGP has reasonable errors but Table 3 shows that the predictive uncertainty of the DeepGP method does not fit the actual uncertainty of the prediction (it has poor coverage probability, either too large or too small). The GPBNN method has

the smallest error (best Q^2_{T}) and it is predicting its accuracy precisely (it has good and nominal coverage probability; here $S = 5$) and the predictive variance and prediction interval width are small compared to the other methods: The GP 1F and AR(1) models have reasonable coverage probabilities but large mean predictive interval widths. For this simple Illustrative example the GPBNN method seems to be the most suitable method.

Table 2: Q^2_{T} for different methods and segments $\bar{I}$ of missing low-fidelity values. Here $S = 5$.

$\bar{I}$	GP 1F	AR(1)	DeepGP	MBK	GPBNN
$\emptyset$	0.12	-0.29	**0.99**	**0.99**	**0.99**
$\left[\frac{1}{3}, \frac{2}{3}\right]$	0.13	-0.34	**0.98**	0.90	**0.98**
$\left[\frac{3}{4}, 1\right]$	0.12	-0.29	0.90	0.51	**0.93**

5.2 2D function approximation

The CURRIN function is a two-dimensional function, with $\mathbf{x} \in [0, 1]^2$. This function is commonly used to simulate computer experiments [3]. The high- and low-fidelity functions are:

$$f_H(\mathbf{x}) = \left[1 - \exp\left(-\frac{1}{2x_2}\right)\right] \frac{2300x_1^3+1900x_1^2+2092x_1+60}{100x_1^3+500x_1^2+4x_1+20}, \tag{42}$$

$$\begin{aligned} f_L(\mathbf{x}) = \; & \tfrac{1}{4}\left[f_H(x_1+\delta, x_2+\delta) + f_H(x_1+\delta, \max(0, x_2-\delta))\right] \\ & + \tfrac{1}{4}\left[f_H(x_1-\delta, x_2+\delta) + f_H(x_1-\delta, \max(0, x_2-\delta))\right], \end{aligned} \tag{43}$$

Table 3: Coverage probability CP_α and mean predictive interval width MPIW_α (between square brackets) for $\alpha = 80\%$ and for different methods and segments $\bar{I}$ of missing low-fidelity values. Here $S = 5$.

$\bar{I}$	GP 1F	AR(1)	DeepGP	MBK	GPBNN
$\emptyset$	**0.82**[0.44]	**0.82**[0.55]	0.99[**0.002**]	0.76[0.037]	**0.88**[**0.083**]
$\left[\frac{1}{3}, \frac{2}{3}\right]$	**0.78**[0.42]	**0.79**[0.45]	0.60[**0.010**]	0.84[0.36]	**0.83**[**0.082**]
$\left[\frac{3}{4}, 1\right]$	**0.78**[0.44]	**0.82**[0.45]	0.62[0.097]	0.86[0.31]	**0.78**[**0.084**]

with $\mathbf{x} = [x_1, x_2]$ and δ the filter parameter. In [3] we have $\delta = 0.05$ and this gives very small differences between the two functions and the prediction of the high-fidelity function by the low-fidelity one has $Q^2_{l\to h} = 0.98$. In the following we set $\delta = 0.1$, and then $Q^2_{l\to h} = 0.87$. An additive Gaussian noise is added to the low-fidelity code. The noise has a zero mean and a variance equal to the empirical variance of the signal 0.08.

In this example we also consider that the low-fidelity code is costly and we only have a small number of low-fidelity points: $N_L = 25$ and $N_H = 15$. The high- and low-fidelity points are chosen by maximin Latin Hypercube Sampling (LHS). The test set is composed of 1000 independent points following a random uniform law on $[0, 1]^2$.

The kernel used for GP regression low-fidelity is a Matérn 5/2 covariance function. The predictive error for the GP regressor of the low-fidelity model is $Q^2 = 0.91$ using a nugget effect in the Gaussian process regression. The BNN is defined with $N_l = 40$ neurons and the mean and variance are evaluated by eq. (36) and corollary 13 with $N_v = 500$. N_l could be chosen arbitrarily but the fact that we are in a small data context leads us to choose a small value. It is possible to use cross-validation to choose N_l, but the computer cost would be here prohibitive. The previous example discussed in section 5.1 suggests choosing S between 3 and 5 for the low-fidelity surrogate model sampling. In this example, due to the large low-fidelity error the model needs a large value of S to account appropriately for the uncertainty and we choose $S = 5$.

All methods have been compared in table 4. The GPBNN model in this example seems to be accurate in terms of Q^2_{T} and in uncertainty quantification. It is much better than the GP 1F model (single-fidelity GP model built with the high-fidelity data). We presume that it is due to the lack of high-fidelity data. AR(1) model gives better but not satisfying results. This is expected due to the non-linearity between codes. The results of the DeepGP method and the MBK method are worse than the one of the GP 1F in error and in uncertainty. For the DeepGP this can be understood by the fact that the covariance is not well adapted, see [3]. And for the MBK the lack of point leads to a very poor optimisation of the hyper-paremeters.

5.3 Double pendulum

The system can be seen as a dual-oscillator cluster, see Figure 5. The system is presented in [20]. The inputs and there variations are presented in [11]. The inputs of the system are of dimension 5, including $(k, M, \theta, \dot{\theta}, y_0)$. The output is of dimension 1, it is the maximum in the axis y of the mass m in the first 10 seconds. We have two codes: the high-fidelity code numerically solves Newton's equation.

Table 4: Comparison of the multi-fidelity methods on the CURRIN function via Q^2_{T}, $\mathrm{CP}_{80\%}$ and $\mathrm{MPIW}_{80\%}$.

	GP 1F	AR(1)	DeepGP	MBK	GPBNN
Q^2_{T}	0.73	0.80	0.29	0.27	0.88
$\mathrm{CP}_{80\%}$	0.68	0.80	0.62	0.57	0.80
$\mathrm{MPIW}_{80\%}$	0.5	1.0	0.13	1.9	0.51

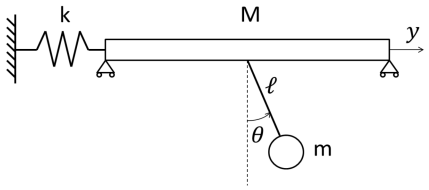


Figure 5: Illustration of the double pendulum.

The low-fidelity code simplifies the equation, by linearisation for small angles of the pendulum motion, and solves the system. Our goal is to build a surrogate model of the high-fidelity code using $N_L = 100$ and $N_H = 20$, with maximin LHS sampling. The input parameters are the mass $M \in [10, 12]$, the spring stiffness $k \in [1, 1.4]$, the initial angle of the pendulum $\theta_0 \in [\frac{\pi}{4}, \frac{\pi}{3}]$, the initial derivative $\dot{\theta_0} \in [0, \frac{1}{10}]$ and the initial position of the mass $y_0 \in [0, 0.2]$. The fixed parameters are $\dot{y_0} = 0$, the gravitational acceleration $g = 9.81$, the length of the pendulum $l = 2$ and its mass $m = 0.5$. The output is the maximum in time of the amplitude of the mass m. The error is computed on a test set, different for each learning set, of 64 samples uniformly distributed on the input space. To evaluate each model we use 5 independent learning and test sets.

The Q^2 for the low-fidelity surrogate model with a Martérn 5/2 as kernel for the GP is 0.98. The BNN is defined with $N_l = 30$ and $N_v = 500$, and we use the GPBNN strategy with $S = 5$. The results are presented in Table 5. The prediction of the MBK method is not accurate compared to all the other methods. We think that this is due to the small data set regarding the dimension of the BNN's input. However, the uncertainty of prediction is still accurate even if the uncertainty interval is large compared to the other methods. This result is very surprising for us in regard with the poor quality of the low-fidelity model, that has

Table 5: Comparison of the multi-fidelity methods on the pendulum example via Q^2_{T}, $\mathrm{CP}_{80\%}$ and $\mathrm{MPIW}_{80\%}$.

	GP 1F	AR(1)	DeepGP	MBK	GPBNN
Q^2_{T}	0.93	0.94	**0.95**	0.54	**0.95**
$\mathrm{CP}_{80\%}$	**0.81**	**0.78**	0.62	0.88	**0.80**
$\mathrm{MPIW}_{80\%}$	0.154	0.146	0.069	0.859	**0.101**

a $Q^2_{l\to l}$ of 0.7. The DeepGP model shares the best predictive error with GPBNN. The single fidelity and the AR(1) models display slightly larger errors. The $\mathrm{CP}_{80\%}$ values are in the target area for GP1F and AR(1) but they are associated with large prediction intervals. The DeepGP clearly underestimates the uncertainty of its predictions. $\mathrm{CP}_{80\%}$ value is acceptable for the GPBNN and close to the target value. Moreover, the uncertainty interval is the smallest of all methods. On this real life example our strategy is competitive compared to other state-of-the-art methods.

6 Conclusion

Our main focus in this paper is to give the Gaussian Process regression posterior distribution of a low-fidelity model as input to a Bayesian neural network for multi-fidelity regression. Deterministic and stochastic methods are proposed and studied to transfer the uncertain predictions of the low-fidelity emulator to the high-fidelity one, which is crucial to obtain minimal predictive errors and accurate predictive uncertainty quantification. The Gauss-Hermite quadrature method is shown to significantly improve the predictive properties of the BNN. The conducted experiments show that the GPBNN method is able to process noisy and real life problems. Moreover, the comparison with some stat-of-the-art methods for multi-fidelity surrogate model highlight the precision in prediction and in uncertainty quantification.

The interest of combining regression method for multi-fidelity surrogate modelling is not to be proved, but this paper adds the heterogeneity of models for multi-fidelity modelling. To be able to combine heterogeneous models into one model and to consider the uncertainty between them is one of the keys to adapt the multi-fidelity surrogate model to a real-life regression problem.

The number of elements in the learning set is not a problem any more thanks

to [22]. We have been able to use many points in the learning set. This approach could be used for the GP part of the GPBNN. With more time in the training the BNN part will be able to deal with many points, even if this ability is less critical because $N_L \gg N_H$. Consequently, the GPBNN can be extended in order to tackle larger data sets.

Existing autoregressive models and Deep GP can only be used for low-dimensional outputs. We wish to extend the method to high-dimensional outputs. Dimension reduction techniques have already been used as principal component analysis or autoencoder, as well as tensorized covariance methods [20] that remain to be extended to the multi-fidelity context. However, neural networks are known to adapt to high-dimensional outputs. We should further investigate how to build multi-fidelity surrogate models with functional input/output in the context of small data. The ability to construct models that are tractable in high dimensions input and output is key for further research.

References

[1] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

[2] H. K. Cigizoglu and M. Alp. Generalized regression neural network in modelling river sediment yield. *Advances in Engineering Software*, 37(2):63–68, 2006.

[3] K. Cutajar, M. Pullin, A. Damianou, N. Lawrence, and J. González. Deep gaussian processes for multi-fidelity modeling. *arXiv preprint arXiv:1903.07320v1*, 2019.

[4] A. I. Forrester, A. Sóbester, and A. J. Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the royal society a: mathematical, physical and engineering sciences*, 463(2088):3251–3269, 2007.

[5] W. Gautschi. Numerical differentiation and integration. In *Numerical Analysis*, pages 159–251. Springer, 2012.

[6] GPy. GPy: A Gaussian process framework in python. `http://github.com/SheffieldML/GPy`, 2012.

[7] M. D. Hoffman and A. Gelman. The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[8] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun. Hands-on Bayesian neural networks–a tutorial for deep learning users. *arXiv preprint arXiv:2007.06823*, 2020.

[9] H. M. D. Kabir, A. Khosravi, M. A. Hosen, and S. Nahavandi. Neural network-based uncertainty quantification: A survey of methodologies and applications. *IEEE Access*, 6:36218–36234, 2018.

[10] M. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 03 2000.

[11] B. Kerleguer. Multi-fidelity surrogate modeling for time-series outputs. *arXiv preprint arXiv:2109.11374*, 2021.

[12] L. Le Gratiet and J. Garnier. Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 4(5):364–386, 2014.

[13] S. Li, W. Xing, R. Kirby, and S. Zhe. Multi-fidelity Bayesian optimization via deep neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 8521–8531. Curran Associates, Inc., 2020.

[14] D. J. MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[15] X. Meng, H. Babaee, and G. E. Karniadakis. Multi-fidelity Bayesian Neural Network: Algorithms and applications. *arXiv preprint arXiv:2012.13294*, 2020.

[16] 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*, 401:109020, 2020.

[17] L. W.-T. Ng and M. Eldred. Multifidelity uncertainty quantification using non-intrusive polynomial chaos and stochastic collocation. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*, page 1852, 2012.

[18] A. O'Hagan. A markov property for covariance structures. *Statistics Research Report*, 98(13):510, 1998.

[19] P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, and G. E. Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2198):20160751, 2017.

[20] G. Perrin. Adaptive calibration of a computer code with time-series output. *Reliability Engineering and System Safety*, 196:106728, 2020.

[21] G. Pilania, J. E. Gubernatis, and T. Lookman. Multi-fidelity machine learning models for accurate bandgap predictions of solids. *Computational Materials Science*, 129:156–163, 2017.

[22] D. Rullière, N. Durrande, F. Bachoc, and C. Chevalier. Nested kriging predictions for datasets with a large number of observations. *Statistics and Computing*, 28(4):849–867, 2018.

[23] T. J. Santner, B. J. Williams, W. Notz, and B. J. Williams. *The design and analysis of computer experiments*. Springer, New York, NY, 2003.

[24] J. Song, Y. Chen, and Y. Yue. A general framework for multi-fidelity bayesian optimization with gaussian processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3158–3167. PMLR, 2019.

[25] R. K. Tripathy and I. Bilionis. Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of Computational Physics*, 375:565–588, 2018.

[26] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

[27] X. Zhang, F. Xie, T. Ji, Z. Zhu, and Y. Zheng. Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 373:113485, 2021.