Multi-fidelity surrogate model combining Gaussian process regressi
and Bayesian neural network

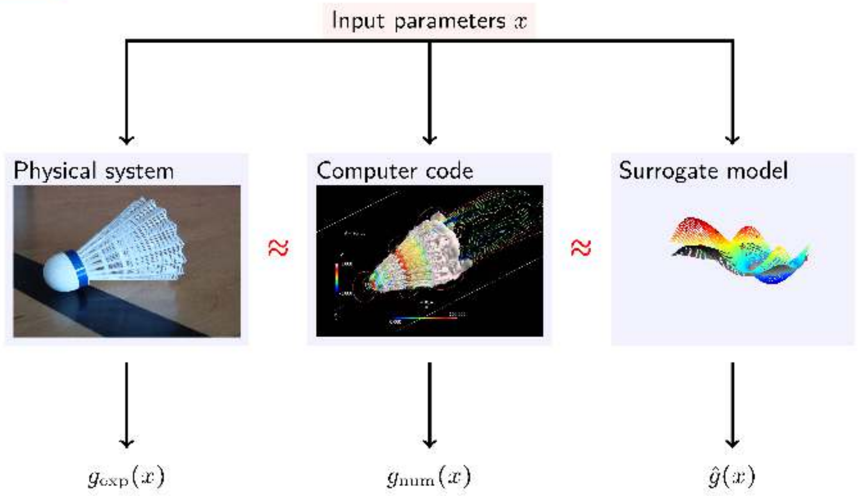
Baptiste Kerleguer[1,2], Claire Cannamela[1], Josselin Garnier[2]

"ETICS 2021" | septembre 2021

[1] CEA,DAM,DIF, F-91297, Arpajon, France,
[2] Centre de Mathématiques Appliquées, Ecole Polytechnique, 91128 Palaiseau Cedex, France

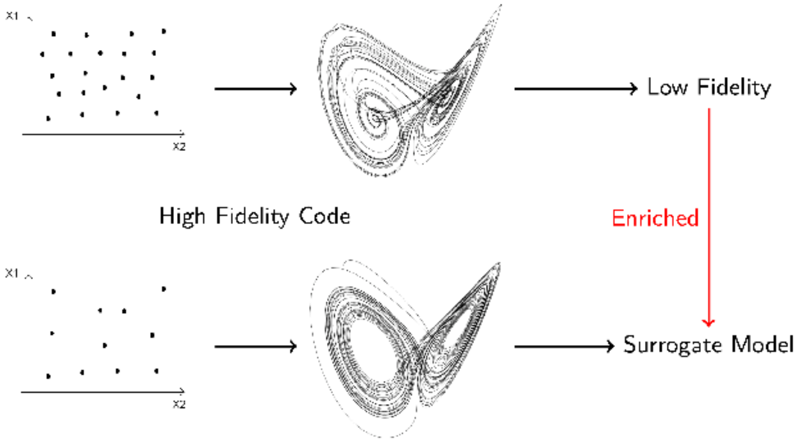
Outline

Model


Input parameters x
Physical system
Computer code
Surrogate model
$\approx$
$\approx$
$g_{\text{exp}}(x)$
$g_{\text{num}}(x)$
$\hat{g}(x)$

Multi-fidelity surrogate model

Low Fidelity Code



High Fidelity Code

Multi-fidelity surrogate model with time-series output

Goals :

- We have high-fidelity and low-fidelity data. We want to generate a surrogate model for the high-fidelity code.
- The interaction between code can be non-linear
- Quantified uncertainties and in particular the high fidelity uncertainty.

Gaussian vector

We want to predict $gX^\star$ knowing the data $\boldsymbol{X}$

- **Gaussian conditioning theorem** Let $(\boldsymbol{X}^\star, \boldsymbol{X} = (X_1, \cdots, X_n))$ be a Gaussian vector of the data and the point to evaluate :

$$\begin{pmatrix} \boldsymbol{X}^\star \\ \boldsymbol{X} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_{\boldsymbol{X}^\star} \\ \boldsymbol{\mu}_{\boldsymbol{X}} \end{pmatrix}, \begin{pmatrix} C_{\boldsymbol{X}^\star} & C_{\boldsymbol{X}^\star\boldsymbol{X}} \\ C_{\boldsymbol{X}\boldsymbol{X}^\star} & C_{\boldsymbol{X}} \end{pmatrix}\right)$$

with $\boldsymbol{\mu}_{gX^\star}$ et $\boldsymbol{\mu}_{\boldsymbol{X}}$ mean vectors of size n and m, the matrix of covariance $C_{\boldsymbol{X}^\star}$ of size $n \times n$, $C_{\boldsymbol{X}^\star\boldsymbol{X}}$ of size $n \times m$, $C_{\boldsymbol{X}^\star\boldsymbol{X}} = C^T_{\boldsymbol{X}^\star\boldsymbol{X}}$ and $C_{\boldsymbol{X}}$ of size $m \times m$.

Then the law of $\boldsymbol{X}^\star$ **conditionally** to $\boldsymbol{X}$ is also Gaussian :

$$(\boldsymbol{X}^\star \mid \boldsymbol{X} = \boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}^{\text{Cond}}(\boldsymbol{x}), C^{\text{Cond}}(\boldsymbol{x})),$$

$$\begin{cases} \boldsymbol{\mu}^{\text{Cond}}(\boldsymbol{x}) = \mathbb{E}(\boldsymbol{X}^\star \mid \boldsymbol{X} = \boldsymbol{x}) = \boldsymbol{\mu}_{\boldsymbol{X}^\star} + C_{\boldsymbol{X}\boldsymbol{Z}} C_{\boldsymbol{X}}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_{\boldsymbol{X}}), \\ C^{\text{Cond}}(\boldsymbol{x}) = C_{\boldsymbol{X}^\star} - C_{\boldsymbol{X}\boldsymbol{X}^\star} C_{\boldsymbol{X}}^{-1} C^T_{\boldsymbol{X}\boldsymbol{X}^\star}. \end{cases}$$

Bayesian Neural Network

We present a BNN with one hidden layer. Let N_n be the number of neurones in the hidden layer. The output of the hidden layer :

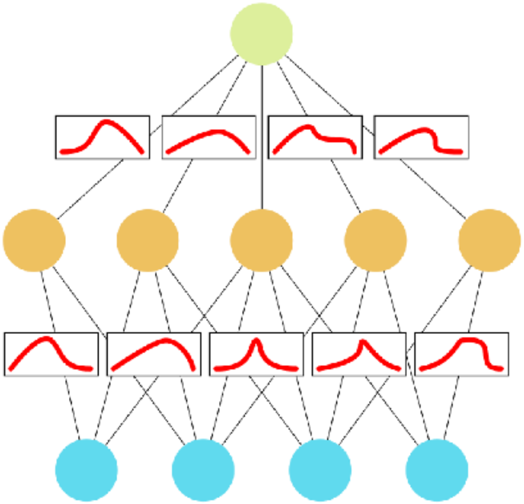
$$\mathbf{y}_1 = \Phi\left(\mathbf{w}_1 \mathbf{x} + \mathbf{b}_1\right), \tag{1}$$

with $\mathbf{x} \in \mathbb{R}^d$ the input vector of the BNN, $\mathbf{b}_1 \in \mathbb{R}^{N_n}$ the bias vector, $\mathbf{w}_1 \in \mathbb{R}^{N_n \times d}$the weight matrix and $\mathbf{y}_1 \in \mathbb{R}^{N_n}$ the output of the hidden layer. The activation function $\Phi : \mathbb{R}^{N_n} \to \mathbb{R}^{N_n}$ is of the form $\Phi\left(\mathbf{b}\right) = \left(\phi(b_i)\right)_{i=1}^{N}$, where ϕ can be hyperbolic tangent or ReLU. The second (and last) layer is fully linear :

$$\mathrm{BNN}(\mathbf{x}) = \mathbf{w}_2^T \mathbf{y}_1 + \mathbf{b}_2, \tag{2}$$

with $\mathbf{w}_2 \in \mathbb{R}^{N_n}$ the weight matrix, $\mathbf{b}_2 \in \mathbb{R}$ the bias vector and the BNN output $\mathrm{BNN}(\mathbf{x}) \in \mathbb{R}$ is the estimation of the output of the function f at a point $\mathbf{x}$. Let the parameter of the BNN $\theta = \left(\mathbf{w}_i, \mathbf{b}_i\right)_{i=1,2}$.

Bayesian Neural Network



Bayesian Neural Network

The pdf of the prediction of y is :

$$p(y|\mathbf{x},\theta,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y-\mathrm{BNN}_\theta(\mathbf{x}))^2}{2\sigma^2}\right), \tag{3}$$

where σ is the standard deviation of the random noise added to account for the fact the neural network is an approximation.
We choose a classic prior distribution for (θ,σ) :

$$\mathbf{w}_i \sim \mathcal{N}\left(\mathbf{0},\sigma_{w_i}^2\mathbf{I}\right), \quad \mathbf{b}_i \sim \mathcal{N}\left(\mathbf{0},\sigma_{b_i}^2\mathbf{I}\right), \quad i=1,2, \quad \sigma \sim \mathcal{N}(0,1), \tag{4}$$

Applying Bayes' theorem the posterior pdf of (θ,σ) given the data $\mathcal{D} = (\mathbf{x}_i,y_i)_{i=1}^n$ is :

$$p(\theta,\sigma|\mathcal{D}) = \prod_{i=1}^{N} p(y_i|\mathbf{x}_i,\theta,\sigma)p(\theta,\sigma) \tag{5}$$

up to a multiplicative constant. The posterior distribution of the output at $\mathbf{x}$ has pdf :

$$p(y|\mathbf{x},\mathcal{D}) = \iint p(y|\mathbf{x},\theta,\sigma)p(\theta,\sigma|\mathcal{D})d\theta d\sigma. \tag{6}$$

Bayesian Neural Network

- Optimization the hyper-parameters :
  - We have the prior distribution of $\mathbf{w}$, $\mathbf{b}$ et σ.
  - An Hamiltonian Monté-Carlo optimization is used to compute the posterior distribution of the hyper-parameters using the available data.
- How to generate the prediction :
  - The problem we get with BNN is that the output depend on different complex law for all parameters. It is impossible to have an analytical expression of the output law depending on the inputs.
  - To tackle this issue we need to sample each law and compute the network with all samples.

Multi-fidelity with GP-BNN

- Objective : model a multi-fidelity system
- Idea : Model the low-fidelity code by a Gaussian process and the high-fidelity code by a bayesian neural network (BNN).

- Decomposition of the problem into two parts :

$$\tilde{f}_L(\mathbf{x}) \sim \mathcal{GP}\left(f_L|\text{data}\right)$$

$$\tilde{f}_H(\mathbf{x}) = \text{BNN}(\mathbf{x}, \tilde{f}_L(\mathbf{x}))$$

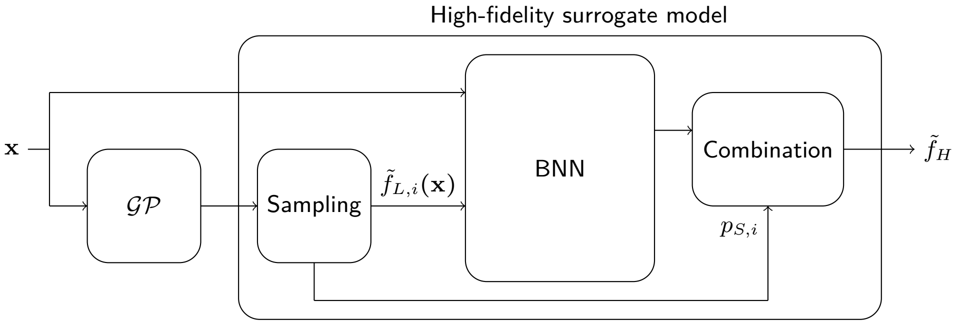


FIGURE – Schematic of the multi-fidelity model.

Sampling of $\tilde{f}_{L,i}(\mathbf{x})$

- Let $z_{S,i}$ be the roots 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 covariance $C_L(\mathbf{x}, \mathbf{x})$.
- Therefore, the ith realization in the Gauss-Hermite quadrature formula is :

$$\tilde{f}_{L,i}(\mathbf{x}) = \mu_L(\mathbf{x}) + z_{S,i}\sqrt{C_L(\mathbf{x}, \mathbf{x})}, \tag{7}$$

- the associated weight is $p_{S,i} = \frac{2^{S-1} S! \sqrt{\pi}}{S^2 H_{S-1}^2(z_{S,i})}$, for $i = 1, \cdots, S$.

The High-fidelity BNN

- The inputs : $\mathbf{x}$ and $\tilde{f}_{L,i}(\mathbf{x})$
- Output : $BNN_{\boldsymbol{\theta}}(\mathbf{x}, \tilde{f}_{L,i}(\mathbf{x}))$
- The estimator of the predictive mean of the output of the high-fidelity model is :

$$\tilde{f}_H(\mathbf{x}) = \frac{1}{N_v} \sum_{j=1}^{N_v} \sum_{i=1}^{S} p_{S,i} BNN_{\boldsymbol{\theta}_j}(\mathbf{x}, \tilde{f}_{L,i}(\mathbf{x})), \tag{8}$$

and the estimator of the predictive variance is :

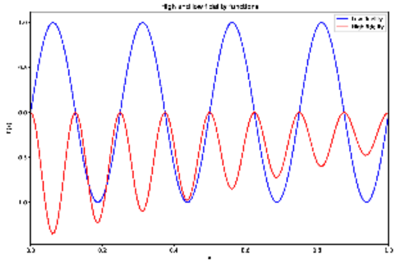
$$\tilde{C}_H(\mathbf{x}) = \frac{1}{N_v} \sum_{j=1}^{N_v} \left(\sum_{i=1}^{S} p_{S,i} BNN_{\boldsymbol{\theta}_j}(\mathbf{x}, \tilde{f}_{L,i}(\mathbf{x})) \right)^2 - \tilde{f}_H^2(\mathbf{x}) + \frac{1}{N_v} \sum_{j=1}^{N_v} \left(\sum_{i=1}^{S} p_{S,i}^2 \right) \sigma_j^2. \tag{9}$$

Optimisation part

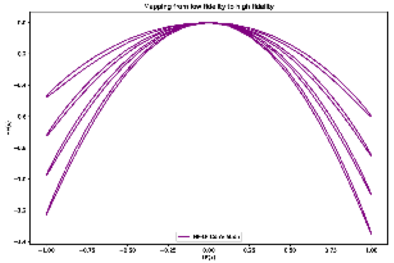
- Step 1, Gaussian process regression for low-fidelity
- Step 2, Sampling data with Gauss quadrature for high-fidelity input
- Step 3, Training the BNN with sampled data
- Step 4, Sampling the all GP-BNN and get uncertainty quantification
- The evaluation of S depending on the problem.

A 1 dimension multi-fidelity function

$$f_L(x) = \sin 8\pi x, \qquad f_H(x) = (x - \sqrt{2}) f_L^2(x), \tag{10}$$

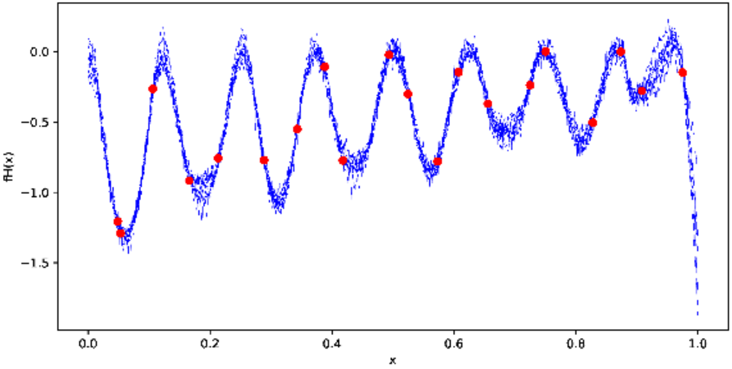


The output of the function High and Low fidelity

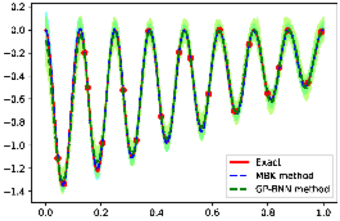


The interaction between codes

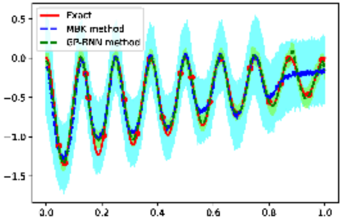
High-fidelity Posterior sampling



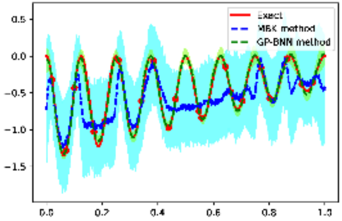
Compareason with BNN only method



(a) $I = \varnothing$



(b) $I = [\frac{3}{4}, 1]$



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

Compareason with known methods

- We evaluate the error Q^2 is $Q^2_{\mathrm{T}} = 1 - \frac{\sum_{i=1}^{N_{\mathrm{T}}}[\tilde{f}_H(\mathbf{x}_{\mathrm{T}}^{(i)}) - f_H(\mathbf{x}_{\mathrm{T}}^{(i)})]^2}{N_{\mathrm{T}}\mathbb{V}_{\mathrm{T}}(f_H)}$.

- For the Methods :
  - Simple fidelity Gaussian Process regression. (GP 1F)
  - Auto-Regressive Multi-fidelity Co-Kriging. (AR(1))
  - Multi-fidelity Deep Gaussian Process (Deep GP)
  - Multi-Fidelity combination of Neural Network and Bayesian Neural Network (MBK)
  - Our method (GP-BNN)

TABLE – Q^2_{T} for different methods and segments $\bar{I}$ of missing low-fidelity values.

$\bar{I}$	GP 1F	AR(1)	Deep GP	MBK	GP-BNN
$\varnothing$	0.12	−0.29	0.99	0.99	0.99
$\left[\frac{3}{4}, 1\right]$	0.12	−0.29	0.93	0.90	0.99
$\left[\frac{1}{3}, \frac{2}{3}\right]$	0.13	−0.34	0.98	0.90	0.98

Compareason with known methods

- We evaluate the Coverage Probability at α (evaluation of the uncertainty accuracy) is

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

And The mean predictive interval at α (size of the uncertainty) is

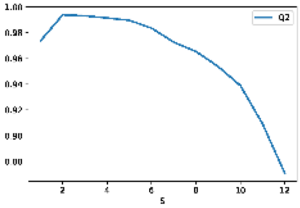
$$\mathrm{MPIW}_\alpha = \frac{1}{N_\mathrm{T}} \sum_{i=1}^{N_\mathrm{T}} \|\mathcal{I}_\alpha(\mathbf{x}_\mathrm{T}^{(i)})\|.$$

- For the Methods :
  - Simple fidelity Gaussian Process regression. (GP 1F)
  - Auto-Regressive Multi-fidelity Co-Kriging. (AR(1))
  - Multi-fidelity Deep Gaussian Process (Deep GP)
  - Multi-Fidelity combination of Neural Network and Bayesian Neural Network (MBK)
  - Our method (GP-BNN)

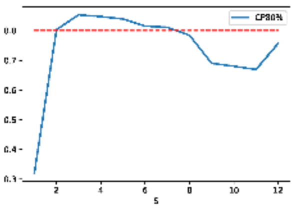
TABLE – Coverage probability CP_α and mean predictive interval width MPIW_α (between the square brackets) for $\alpha = 80\%$ and for different methods and segments $\bar{I}$ of missing low-fidelity values.

$\bar{I}$	GP 1F	AR(1)	Deep GP	MBK	GP-BNN
$\varnothing$	$0.82\,[0.44]$	$0.82\,[0.55]$	$0.99\,[0.002]$	-	$0.88\,[0.083]$
$[\frac{3}{4}, 1]$	$0.78\,[0.44]$	$0.82\,[0.45]$	$0.62\,[0.097]$	-	$0.78\,[0.084]$
$[\frac{1}{3}, \frac{2}{3}]$	$0.78\,[0.42]$	$0.79\,[0.45]$	$0.60\,[0.010]$	-	$0.83\,[0.082]$

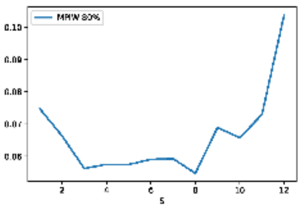
ăStudy of the choice of S



(a) Q^2_{T}



(b) $\mathrm{CP}_{80\%}$



(c) $\mathrm{MPIW}_{80\%}$

FIGURE – Evaluation of the Q^2_{T}, the coverage probability at 80% and $\mathrm{MPIW}_{80\%}$ depending on S.

Perspectives

- Conclusion :
  - The model is efficient in small data
  - Good uncertainty prediction in high-fidelity
- Ongoing :
  - Growing the dimension of output (time-series)
  - Wavelet representation for time-series output
- Future :
  - Structure with full BNN
  - Using a more complex structure for the BNN

Extention Large Dimension

- Replace the 1 dimension GP in GP-BNN by tensorised covariance GP :
  - The Neural Network can solve the curve of dimensionality.
  - but for BNN it is more complex and in particular a good choice of prior will be very useful.
  - The solution is to put train a neural network with a last Bayesian layer.
- Reduce the dimension :
  - If the reduction of dimension is linear then the best model with be the linear model
  - For a non-linear model we train multiple GP-BNN. In particular, we can use autoencoders.
  - The number of parameters to choose will be big.

Result for large dimension

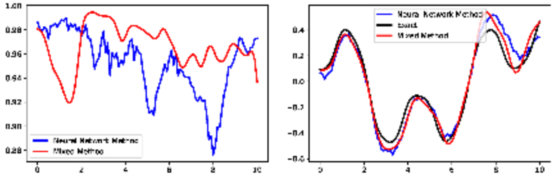


Figure – Evaluation of the Q^2, depending on the output time, compare with the mixed method. And a realization of the output.

- The reduction BNN method is not yet mature. We need to improve the optimisation :

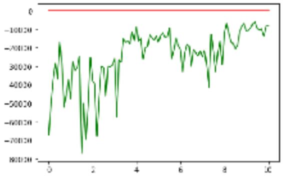


Figure – Evaluation of the Q^2, depending on the output time, compare with the mixed method.

Questions ?