Physics-informed neural networks to propagate random field properties of composite materials

D. Bonnet-Eymard [1], **A. Persoons** [1], **P. Gavallas** [2], **M. GR Faes** [3], **G. Stefanou** [2], **D. Moens** [1]
[1] KU Leuven, Department of Mechanical Engineering,
Jan Pieter de Nayerlaan 5, 2860 Sint-Katelijne-Waver, Belgium
e-mail: **damien.bonnet-eymard@kuleuven.be**

[2] Aristotle University of Thessaloniki, Department of Civil Engineering,
54124 Thessaloniki, Greece

[3] TU Dortmund University, Chair for Reliability Engineering,
Leonhard-Euler-Strasse 5, Dortmund 44227, Germany

Abstract

Composite materials are challenging to use in critical applications due to the difficulty of modeling their randomly varying spatial properties. Polynomial Chaos Expansion (PCE) is a popular surrogate method for propagating such uncertainties. However, its data-based calibration requires potentially expensive evaluations of the reference model, motivating the need for cost efficiency improvements. The recently proposed PINN-PCE approach, a surrogate that combines Physics-Informed Neural Network (PINN) and Polynomial Chaos Expansion, aims at this cost reduction by leveraging physics knowledge of the problem. In this work, we apply the PINN-PCE framework on a 2D composite plate example. Uncertain stiffness coefficients, modeled by a lognormal random fields, are propagated to the material response (i.e. the displacement field). Although the results are currently limited by the computational burden of the method, workarounds are planned for future improvements.

1 Introduction

Due to their complex microstructure, composite materials exhibit properties that vary randomly in space. It is crucial to account for this randomness, as it can significantly impact material response. In their work, Stefanou et al. (2022) [1] proposed a framework to propagate this uncertainty. They modeled the material properties as random fields, calibrated from simulations of the microstructure. The material response was then computed using a Finite Element Method (FEM) solver, and the uncertainty was propagated via Monte Carlo simulations.

This approach, however, can become computationally intractable for complex examples, as it requires a large number of FEM simulations. To alleviate this limitation, surrogate-based approaches such as Polynomial Chaos Expansion (PCE) have been widely used in the literature [2]. Nonetheless, calibrating the surrogate model remains a purely data-based and computationally demanding step, which could be further improved by incorporating physics knowledge.

Zhang et al. (2019) [3] proposed such an approach by combining Physics-Informed Neural Networks (PINN) with PCE to propagate uncertain parameters. They referred to this method as NN-aPC (Neural Network-arbitrary Polynomial Chaos). Arbitrary Polynomial Chaos (aPC) is a generalization of PCE that can handle arbitrary probability distributions. However, since all distributions in this work will be standard, we will not utilize aPC. For this reason and to emphasize the incorporation of physics knowledge, we will refer to this approach as PINN-PC hereafter.

In their work, Zhang et al. mainly focused on inverse problems, seeing here the most potential for their approach. Our work, on the other hand, aims to address a more complex and realistic forward problem.

After briefly introducing both PINN and PCE, we will detail the PINN-PC framework. The framework is then applied to the Poisson equation as a validation step, to finally address a 2D composite plate example.

Related work

Physics-constrained Polynomial Chaos Expansion [4, 5] (PC^2) is another recent approach aiming at adding physics knowledge to the PCE framework. This approach is different from the PINN-PC presented here, as it does not use neural networks. A benchmark study of the two approaches is envisaged as future work.

2 Methodology

2.1 Physics-Informed Neural Networks (PINNs)

Given the extensive body of research and numerous publications on the topic, only a brief introduction to Physics-Informed Neural Networks (PINNs) is provided here. For more comprehensive details, the reader is referred to the original paper by Raissi et al. (2019) and the literature review by Cuomo et al. (2022) [6, 7].

Physics-Informed Neural Networks (PINNs) are a class of neural networks designed to approximate solutions to boundary value problems (BVPs) defined by a partial differential equations (PDE) and boundary conditions:

$$\begin{cases} \mathcal{F}_\lambda[u](x) = f(x) & \text{in} \quad \Omega, \\ \mathcal{B}_\lambda[u](x) = g(x) & \text{on} \quad \partial\Omega, \end{cases} \tag{1}$$

where $\mathcal{F}_\lambda$ is the PDE operator, $\mathcal{B}_\lambda$ is the boundary operator, u is the solution, f is the source term, g is the boundary condition, and Ω is the domain.

The solution u is approximated by a neural network u_θ with weights θ, trained to minimize a loss function $\mathcal{L}_{\text{PINNs}}$ defined as:

$$\mathcal{L}_{\text{PINNs}} = \mathcal{L}_{\text{physics}} + \mathcal{L}_{\text{IC/BC}} + \mathcal{L}_{\text{data}}, \tag{2}$$

where $\mathcal{L}_{\text{physics}}$ enforces the PDE, $\mathcal{L}_{\text{IC/BC}}$ enforces the initial and boundary conditions, and $\mathcal{L}_{\text{data}}$ enforces compliance with observed data.

Initial and boundary conditions can also be enforced using a masking function that multiplies the neural network output, as proposed by Lagaris et al. (1998) [8]. This approach ensures that the approximation inherently satisfies the boundary conditions, referred to as *hard constraints*, as opposed to the *soft constraints* of additional loss terms. While hard constraints can be challenging to implement depending on the problem's geometry, they generally result in better convergence.

In the context of physics problems, available data typically corresponds to measurements. However, data is not mandatory; the solution can be sought using only the PDE and boundary conditions, solving a so-called *forward problem*. Conversely, when data is available, it can be used to find the parameters λ of the PDE, solving an *inverse problem*.

The PINN framework was initially designed to solve deterministic problems. However, Zhang et al. (2019) [3] proposed an extension to handle uncertain parameters using Polynomial Chaos Expansion (PCE).

2.2 Polynomial Chaos Expansion (PCE)

Polynomial Chaos Expansion (PCE) is a popular surrogate modeling technique that has been widely used in uncertainty quantification. This framework will be only briefly introduced here, the interested reader is referred to the UQLab user manual [9] for more details. The goal of PCE is to propagate uncertain input parameters through the output of a model. The main idea is to represent this ouptut as a polynomial

expansion of the random input parameters. Consider a model $\mathbf{Y} = \mathcal{M}(\xi)$ where $\xi \in \mathbb{R}^M$ is a random vector with independent components and $\mathbf{Y}$ is the output of the model. The PCE surrogate model of $\mathcal{M}$ is given by

$$\mathbf{Y} \approx \mathcal{M}^{\text{PCE}}(\xi) = \sum_{\alpha \in \mathcal{A}} y_\alpha \Psi_\alpha(\xi), \tag{3}$$

where $\mathcal{A} \subset \mathbb{N}^M$ is the set of multi-indices, y_α are the PCE coefficients, and $\Psi_\alpha(\xi)$ are the multivariate polynomials. The polynomials are constructed to be orthogonal with respect to a probability distribution of the input parameters (see [9] for more details). The set of multi-indices $\mathcal{A}$ encode the truncation of the expansion, which is a trade-off between accuracy and computational cost. Many strategies exist in the literature to optimally choose this truncation (e.g. sparse PCE [10]). They will not be detailed here since low-order polynomials will be sufficient for this work.

Once the PCE basis functions $\Psi_\alpha(\mathbf{X})$ are defined, calibrating the model consists of finding the PCE coefficients y_α. This step is usually data-based: the model $\mathcal{M}$ is evaluated at a set of points and the PCE coefficients are determined by regression. The idea of Zhang et al. (2019) [3], on the other hand, was to leverage the physics knowledge to improve the calibration of the PCE model.

Classic PCE models, as written in Eq. 3, only take random variables as input. However, in our case, the space coordinates (and eventually time) should also be considered as input. There are different ways to handle space variables in the PCE framework. The main one is to make the coefficients of the PCE model dependent on the space variables: $y_\alpha = y_\alpha(x)$. This is the approach behind PINN-PC, using PINNs to approximate the coefficients $y_\alpha(x)$ of the PCE model.

2.3 PINN-PC

The main idea of the PINN-PC framework is to use PINNs to approximate the coefficients of the PCE model. The model architecture is illustrated in Figure 1.

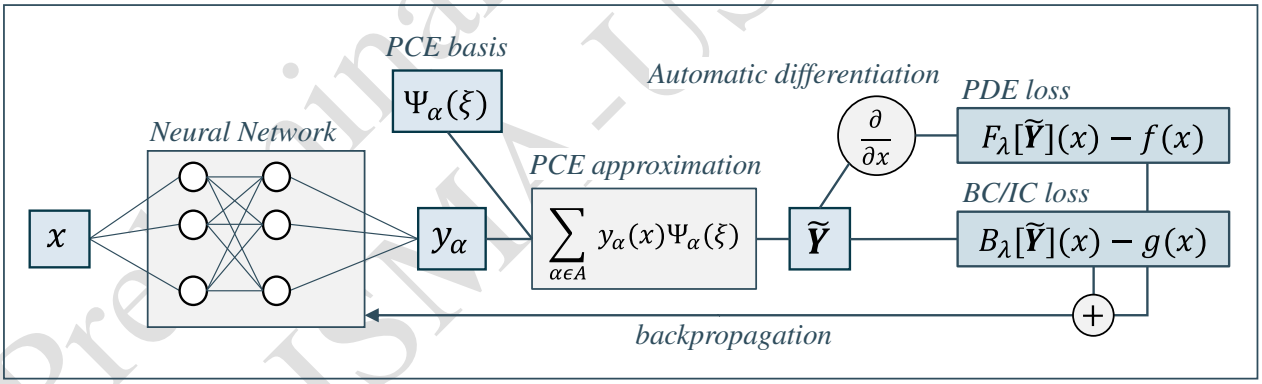


Figure 1: Schematic representation of the PINN-PC framework.

The neural network takes as input the space coordinates x and outputs the space-dependent (and eventually time-dependent) PCE coefficients $y_\alpha(x)$. The random variables are taken into account through the PCE basis functions $\Psi_\alpha(\xi)$. The two are combined using the polynomial expansion formula (Eq. 3), to obtain the output of the model $\mathbf{Y}$, which is a space-dependant random variable expanded in the PCE basis. As all the steps are differentiable, the partial derivatives of the output with respect to the space coordinates can be computed using automatic differentiation. Consequently, like in the PINN framework, the PDE of the approximation can be computed and used as a loss function to train the neural network.

To predict the coefficients $y_\alpha(x)$ with neural networks, various architectural choices can be made. Zhang et al. (2019) [3] proposed grouping the coefficients by their polynomial degree and using a separate neural network for each group, explaining that coefficients of the same degree are expected to have similar magnitudes.

To train the model, a set of M-dimensional samples of random vectors $S := \{\xi_s\}_{s=1}^{N}$ is required to account for the random variables. The size of the set N is a trade-off between accuracy and computational cost as it should be large enough to fairly represent the variables distributions. Hence this framework increases the computational cost by a factor N compared to deterministic PINNs. A possible strategy to reduce this cost is discussed in section 4.

3 Experiments

All the experiments were conducted using the PyTorch library on an NVIDIA Tesla V100 GPU. The code will be made available on GitHub: `www.github.com/bonneted/USD2024`

3.1 Poisson equation

As a first validation step, we reproduce the results of Zhang et al. (2019) [3] on the Poisson equation.

3.1.1 Problem Statement

We consider the 1D stochastic Poisson equation with Dirichlet boundary conditions:

$$\begin{aligned} -\frac{d^2u}{dx^2} &= f(x,\xi), \quad x \in [0,1], \\ u(0) &= u(1) = 0. \end{aligned} \tag{4}$$

The source term $f(x,\xi) \sim \mathcal{GP}(f_0(x), k(x,x'))$ is a Gaussian random field with mean $f_0(x) = 10\sin(2\pi x)$ and a squared exponential covariance function:

$$k(x,x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{l^2}\right), \tag{5}$$

where $\sigma = 1$ and $l = 0.5$.

The first step is to discretize the forcing term $f(x,\xi)$, to construct the polynomial chaos expansion on a finite number of random variables.

3.1.2 Discretization of the forcing term

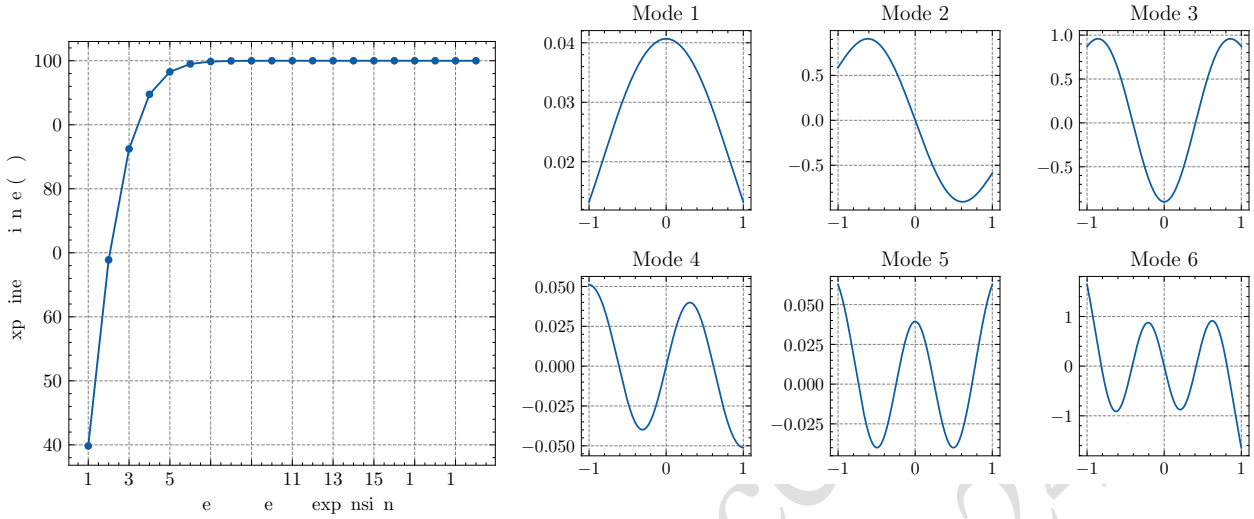
The approach used here differs from the aPC used by Zhang et al. (2019) [3], but only in its formulation, as the two discretizations are strictly equivalent. Indeed, Zhang et al. used principal component analysis (PCA) combined with arbitrary polynomial chaos to construct the basis functions of the PCE. Nonetheless, as the random field is Gaussian, the basis functions of the PCE can be directly constructed using the Karhunen-Loève expansion. In a discrete setting, the K-L expansion is achieved by the eigendecomposition of the covariance matrix of the random field, which is equivalent to the PCA of the samples.

The forcing term is discretized on a grid of $N = 100$ points on the domain $\Omega = [0,1]$. The covariance matrix of the random field is then computed using the covariance function (Eq. 5) and its eigendecomposition is performed. The truncation of the expansion is chosen to keep 99% of the variance (equal to the cumulative sum of the eigenvalues), resulting in $M = 6$ basis functions (see Figure 2a). The first 6 eigenvectors are plotted in Figure 2b.

The K-L expansion can be written as:

$$f(x,\xi) = f_0(x) + \sum_{m=1}^{M} \sqrt{\lambda_m}\xi_m\phi_m(x), \tag{6}$$

where λ_m are the eigenvalues, ξ_m are standard normal random variables, and $\phi_m(x)$ are the eigenfunctions. Hence, the K-L expansion can be seen as a first-order PCE (only the ξ_m are considered) with coefficients $y_\alpha = \sqrt{\lambda_m}\phi_m(x)$.



(a) Explained variance

(b) First eigenvectors

Figure 2: Eigendecomposition of the covariance matrix to construct the K-L expansion.

With this decomposition, it is possible to compute a Monte Carlo reference approximation of the PCE coefficients y_α. For this simple example, we only consider a first-order PCE, i.e. $\psi_\alpha(\xi) = \xi_\alpha$ and $\alpha \in \{1, \dots, M\}$.

We sampled 10^6 random vectors ξ and computed the forcing term $f(x, \xi)$ at each point x of the grid. For each forcing term, the Poisson equation is solved using finite differences and the solution $u(x, \xi)$ is stored. The mean and standard deviation of the solution are plotted in Figure 3.

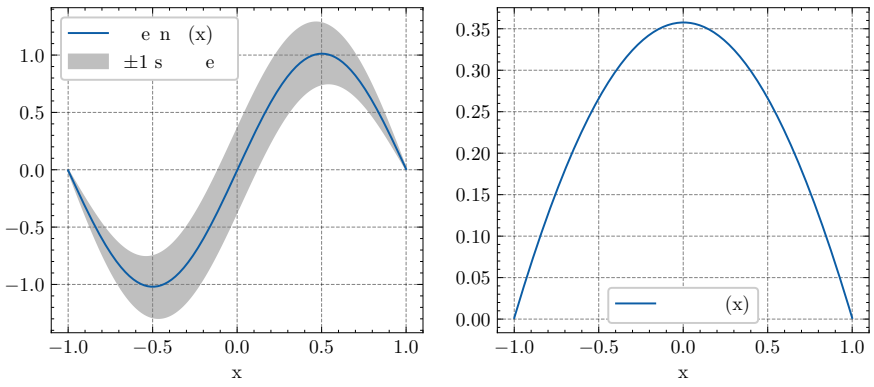


Figure 3: Mean and standard deviation of the solution computed using 10^6 Monte Carlo simulations.

The PCE coefficients y_α can be approximated using the projection method [9] :

$$y_\alpha(x) = \frac{1}{N}\sum_{s=1}^{N} u(x, \xi_s)\Psi_\alpha(\xi_s) = \frac{1}{N}\sum_{s=1}^{N} u(x, \xi_s)\xi_{\alpha,s}, \tag{7}$$

where $\xi_{\alpha,s}$ is the α-th component of the s-th random vector ξ_s.

The resulting reference PCE coefficients are plotted in Figure 4. Those are the space-dependent coefficients we aim to approximate using the PINN-PC framework.

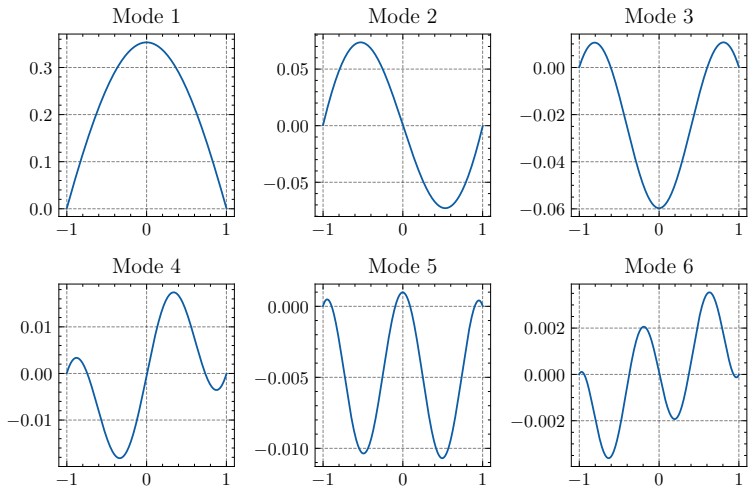


Figure 4: Reference PCE coefficients computed using 10^6 Monte Carlo simulations.

3.1.3 PINN-PC implementation

Following the implementation of Zhang et al. (2019) [3], we used two different multilayer perceptrons (MLPs) with hyperbolic tangent activation functions:

- MeanNN, of size $[1, 4, 4, 1]$, to approximate the mean solution $u_0(x)$,
- CoeffNN, of size $[1, 36, 36, 36, 36, M]$, to approximate the PCE coefficients $y_\alpha(x)$.

The model was trained using the Adam optimizer with a learning rate of 10^{-3} for 20000 iterations. We achieved similar results to Zhang et al. (2019) [3], the model was able to accurately predict the mean and standard deviation of the solution (Figure 5) and the PCE coefficients (Figure 6).

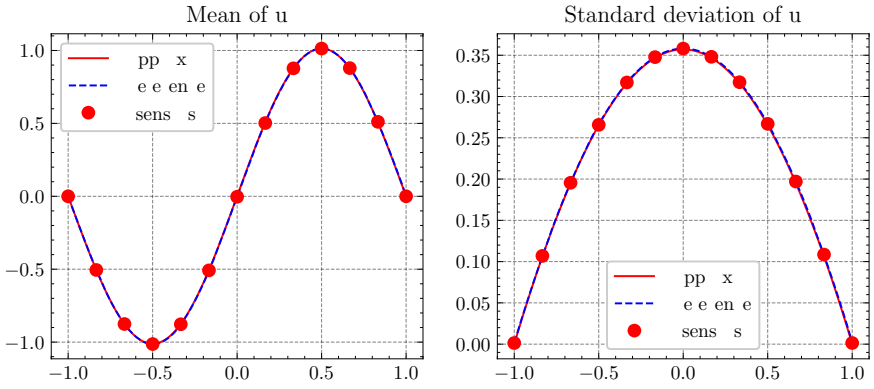


Figure 5: Mean and standard deviation of the solution computed using the PINN-PC framework.

Building on this conclusive first result, we now aim to apply the PINN-PC framework to a more complex example: a 2D composite plate.

3.2 Composite plate

In their work, Stefanou et al. (2022) [1] proposed a framework to propagate the random spatial variations of the stiffness matrix components of a composite plate. Stiffness components are modeled as random fields calibrated from simulations of the microstructure. The material is then subjected to a mechanical load, and the displacement field is computed using a FEM solver. The uncertainty is propagated via Monte Carlo simulations.

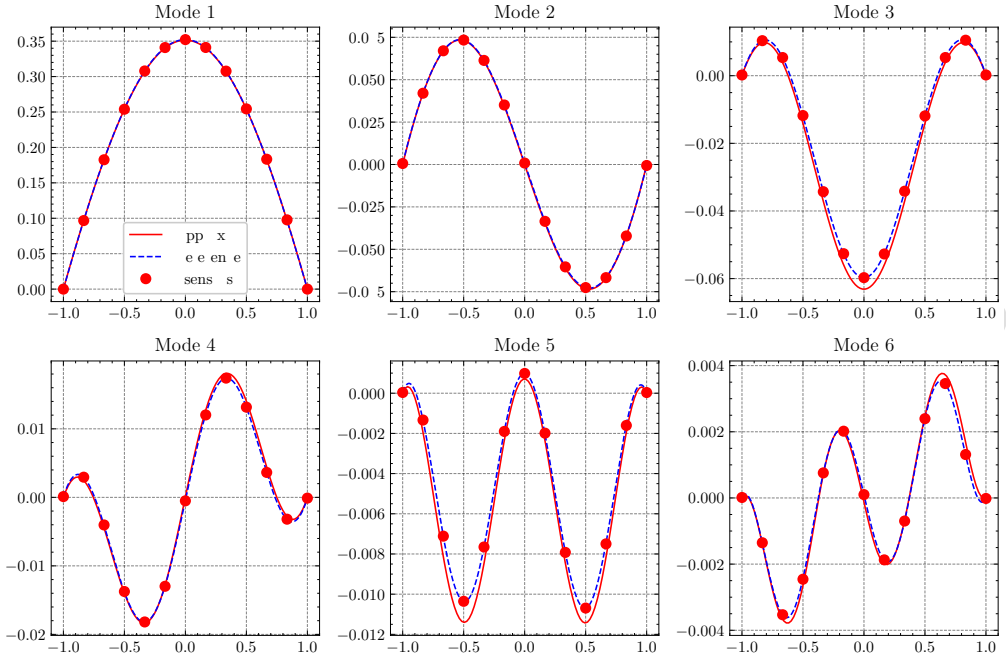


Figure 6: PCE coefficients approximated using the PINN-PC framework.

3.2.1 Problem statement

We consider a 2D composite plate of dimensions $100\mu\text{m} \times 100\mu\text{m}$. A tensile side load $p = 0.1\text{N}/\mu\text{m}$ is applied on the right side of the plate. The boundary conditions are illustrated in Figure 7.

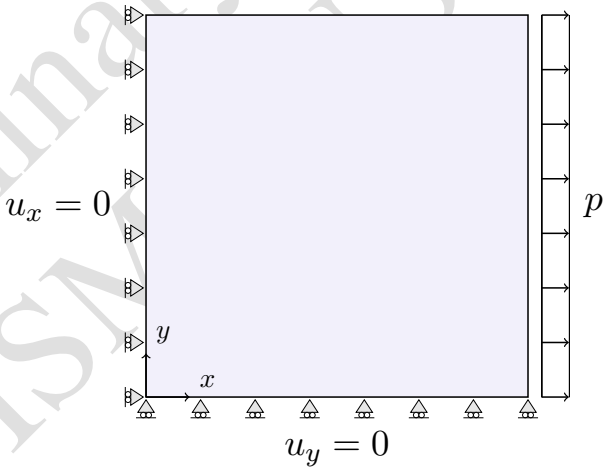


Figure 7: Boundary value problem for the composite plate.

The material is considered to be linear elastic and isotropic. The stiffness matrix can be expressed in terms of stiffness components C_{11} (axial) and C_{33} (shear) as follows:

$$\mathbf{C} = \begin{bmatrix} C_{11} & C_{11} - 2C_{33} & 0 \\ C_{11} - 2C_{33} & C_{11} & 0 \\ 0 & 0 & C_{33} \end{bmatrix}. \tag{8}$$

The stiffness components C_{11} and C_{33} are modeled as lognormal random fields with realistic parameters calibrated from simulations of the microstructure, after Stefanou et al. (2022) [1]. The parameters are summarized in Table 1, L_x and L_y are the correlation lengths in the x and y directions.

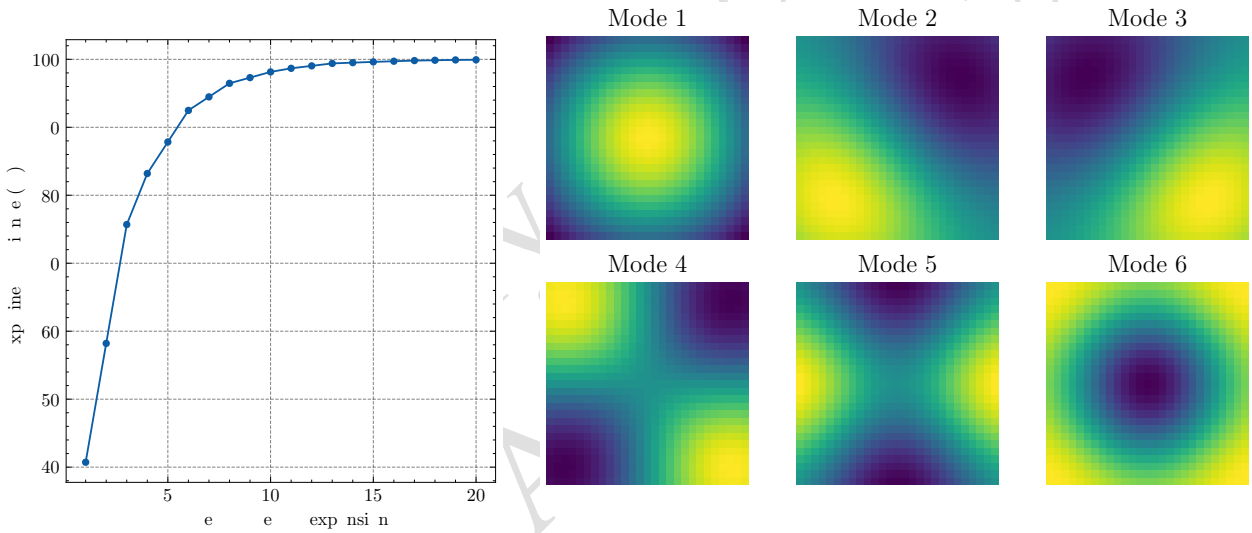
Table 1: Parameters of the underlying Gaussian random fields modeling the stiffness components.

Parameter	Mean	Std	Lx	Ly
C_{11}	1.55	0.129	50	50
C_{33}	0.316	0.027	50	50

These random fields need to be discretized in order to construct the PCE basis functions for the PINN-PC framework.

3.2.2 Modeling the plate properties

Like in the Poisson equation example, the random fields are discretized using the Karhunen-Loève expansion. We choose here a squared exponential covariance function for the random fields to keep the number of basis functions low. The problem is discretized on a grid of $N = 100 \times 100$ points. Eigendecomposition of the covariance matrix is performed, and truncation of the expansion is chosen to keep 99% of the variance, resulting in $M = 8$ basis functions. The explained variance and the first 6 eigenvectors of the C_{11} stiffness component are plotted in Figure 8.



(a) Explained variance

(b) 6 first eigenvectors

Figure 8: Eigendecomposition of the covariance matrix of the C_{11} stiffness component.

Using this decomposition, we can simulate the stiffness components C_{11} and C_{33} and compute the displacement field of the plate using a FEM solver. To set a reference solution, we carried out 10^4 Monte Carlo simulations. The mean and standard deviation of the displacement field are plotted in Figure 9.

3.2.3 PINN-PC implementation

The PINN-PC framework is then applied to approximate the PCE coefficients of the stiffness components. The model architecture is similar to the one used for the Poisson equation, with the following MLPs:

- MeanNN, of size $[2, 4, 4, 1]$, to approximate the mean solution $u_0(x, y)$,
- CoeffNN, of size $[2, 36, 36, 36, 36, 2M]$, to approximate the PCE coefficients $y_\alpha(x, y)$ of the stiffness coefficients.

Despite several hyperparameter tuning attempts, the model was not able to accurately predict the mean and standard deviation of the displacement field. One main limitation is the computational cost of the frame-

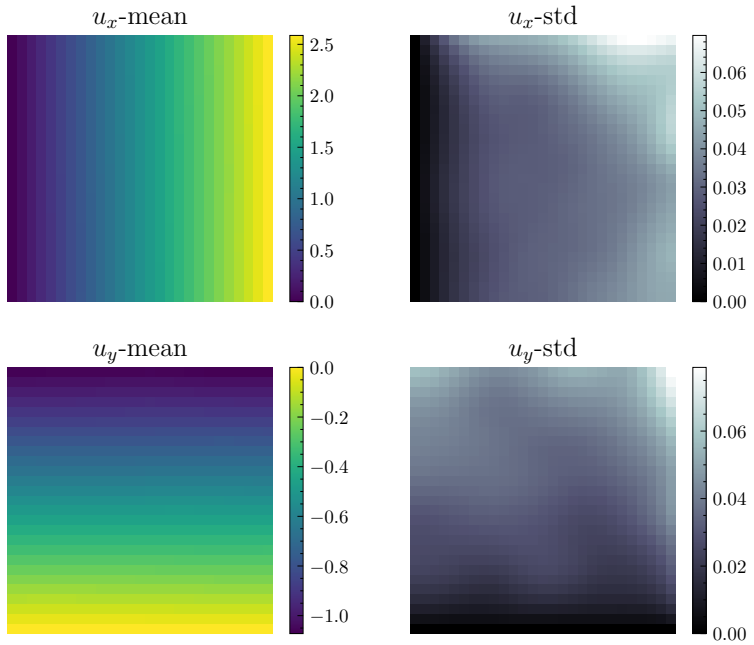


Figure 9: Mean and standard deviation of the displacement field computed using 10^4 Monte Carlo simulations.

work, which increases by a factor N compared to deterministic PINNs. A potential solution to mitigate this limitation is discussed in the next section.

4 Discussion and future work

The PINN-PC framework was successfully applied to a 1D Poisson equation, reproducing the results of Zhang et al. (2019) [3]. The model was able to accurately predict the mean and standard deviation of the solution and the PCE coefficients. The framework was then applied to a 2D composite plate example. Even though every step of the framework was successful, this is still a work in progress and convergence was not achieved. The limiting computational burden of the framework could potentially be alleviated using the recently introduced and cost-effective Separable Physics-Informed Neural Networks (SPINNs) framework [11]. This promising approach will be investigated in future work.

Acknowledgements

The authors gratefully acknowledge the support of the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie project GREYDIENT – Grant Agreement n°955393.

The authors gratefully acknowledge the support of the Flemish Government through the Flanders Make initiative.

References

[1] G. Stefanou, D. Savvas, P. Gavallas, and I. Papaioannou, "The effect of random field parameter uncertainty on the response variability of composite structures," *Composites Part C: Open Access*, vol. 9, p. 100324, Oct. 2022.

[2] B. Sudret, "Polynomial chaos expansions and stochastic finite element methods," in *Risk and Reliability in Geotechnical Engineering*, J. C. Kok-Kwang Phoon, Ed. CRC Press, 2015, pp. 265–300.

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

[4] L. Novák, H. Sharma, and M. D. Shields, "Physics-Informed Polynomial Chaos Expansions," Sep. 2023.

[5] H. Sharma, L. Novák, and M. D. Shields, "Physics-constrained polynomial chaos expansion for scientific machine learning and uncertainty quantification," May 2024.

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

[7] S. Cuomo, V. S. di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next," Jun. 2022.

[8] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, Sep. 1998.

[9] S. Marelli and B. Sudret, *UQLab User Manual - Polynomial Chaos Expansions*, Jul. 2015.

[10] G. Blatman and B. Sudret, "Adaptive sparse polynomial chaos expansion based on least angle regression," *Journal of Computational Physics*, vol. 230, no. 6, pp. 2345–2367, Mar. 2011.

[11] J. Cho, S. Nam, H. Yang, S.-B. Yun, Y. Hong, and E. Park, "Separable Physics-Informed Neural Networks," Oct. 2023.

Appendix

A Nomenclature

PCE	Polynomial Chaos Expansion
aPC	Arbitrary Polynomial Chaos
$PINN$	Physics-Informed Neural Network
$NN\text{-}aPC$	Neural Network-arbitrary Polynomial Chaos
$PINN\text{-}PC$	Physics-Informed Neural Network - Polynomial Chaos Expansion
PCA	Principal Component Analysis
$K\text{-}L$	Karhunen-Loève
GP	Gaussian Process
FEM	Finite Element Method
PDE	Partial Differential Equation
BVP	Boundary Value Problem
IC	Initial Condition
BC	Boundary Condition