Physics-Informed Generative Neural Networks for Stochastic Differential Equations

Lucia Clarotto Hugo Gangloff Nicolas Jouvin Antoine Regardin
MIA-PS, INRAE

Outline

Introduction

- Study of stochastic PDEs using machine learning.
- Focus on Physics-Informed Neural Networks (PINNs).
- Integration of randomness via Polynomial Chaos and Karhunen-Loève expansions.
- Final utility: modelize and infer on random fields

Random Fields

- Random function: $X(s, \omega)$.
- Gaussian Random Field (GRF): finite collections $\sim \mathcal{N}_n(\mu, \Sigma)$.
- Covariance: $C(x, y) = \text{Cov}(Z(x), Z(y))$.
- Stationarity: X is second order stationary if $\text{Var}[Z(x)] < \infty$,

  $\forall s, t, \tau \in \mathbb{R}$, $\mathbb{E}[X(s)] = \mathbb{E}[X(s + \tau)]$,

  Cov(Z(s),Z(t)) = C(s,t) = Cov(X(s + τ), $X(t + \tau)$).

In this case, we can note the auto-covariance $C(h)$ in function of a lag $h = s - t$

Variogram and Correlogram

Variogram of a second-order stationary random field:

$$\gamma(h) = \frac{1}{2}\,\mathbb{E}[(X(s+h) - X(s))^2]$$

We can use the empirical estimator:

$$\hat{\gamma}(h) = \frac{1}{2|N(h)|} \sum_{(i,j)\in N(h)} (Z(x_i) - Z(x_j))^2$$

In practice, we will look at the correlogram:

$$\rho(h) = 1 - \frac{\gamma(h)}{\sigma^2}$$

Hermite Polynomials

Defined by:

$$H_n(x) = (-1)^n e^{x^2/2} \frac{d^n}{dx^n} \left(e^{-x^2/2}\right)$$

Orthogonality:

$$\int H_n(x) H_m(x) \frac{e^{-x^2/2}}{\sqrt{2\pi}} dx = n!\, \delta_{nm}$$

Recurrence:

$$H_{n+1}(x) = xH_n(x) - nH_{n-1}(x)$$

Can be used as a basis for random function spaces.

Polynomial Chaos Expansion (PCE)

Any random variable $X \in L^2(\Omega)$ can be written as

$$X(\omega) = \sum_{\alpha \in \mathbb{N}_0^N} c_\alpha \Psi_\alpha(\boldsymbol{\xi}(\omega))$$

Truncated:

$$X(\omega) \approx \sum_{|\alpha| \leq p} c_\alpha \Psi_\alpha(\boldsymbol{\xi}(\omega))$$

Where $\boldsymbol{\xi}$ is a gaussian vector. We will be using $\Psi_\alpha = H_\alpha$

Karhunen-Loève Expansion

For a random field $X(s, \omega)$ with covariance $C(s, t)$:

$$X(s, \omega) = \sum_{n=1}^{\infty} \sqrt{\lambda_n}\, \phi_n(s)\, \xi_n(\omega)$$

where $\{\lambda_n, \phi_n\}$ are eigenpairs of the covariance operator:

$$\int_D C(s, t)\, \phi_n(t)\, dt = \lambda_n \phi_n(s),$$

The truncated KL expansion is a minimal representation of a Gaussian Random Field.

Stochastic PDE

$$\mathfrak{D}u(t,x) = f(t,x,u) + g(t,x,u)\,W(t,x)$$

- $\mathfrak{D}$: differential operator.
- u: unknown random field.
- W: Random field.

Collocation Methods

Approximation:

$$u_N(x) = \sum_{j=1}^{N} c_j \varphi_j(x)$$

Enforce SPDE at collocation points:

$$\mathfrak{D} u_N(x_i) = f(x_i) + W(x_i)$$

The solve a linear system: $Ac = \mathrm{f} + \boldsymbol{W}$. Such methods are suited for pointwise-defined random fields, but not for distribution-based ones (i.e. solutions of weak forms of problems).

Physics-Informed Neural Networks

Our type of SPDE:

$$\mathfrak{D}u(x) = f(x), \quad x \in D, \quad f \quad \text{a GRF}$$

Loss:

$$\mathcal{L}(\theta) = \frac{1}{N_f} \sum_{j=1}^{N_f} \|\mathfrak{D}[u_\theta(\mathrm{x}_j)] - f(\mathrm{x}_j)\|^2$$

The goal being to replace the system solving of a collocation method by a neural network.
Here, we will use Multi-Layer Perceptrons, with hyperbolic tangent activation function.

Matérn Covariance

A Matérn random field is a second-order GRF with the covariance:

$$C(h) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\|h\|}{\rho}\right)^{\nu} K_{\nu}\left(\frac{\|h\|}{\rho}\right)$$

A Matérn field X is equivalent to an Exponential random field if $\nu = \frac{1}{2}$.

Matérn SPDE

A Matérn random field can be defined as the solution of the SPDE:

$$(\kappa^2 - \Delta)^{\alpha/2} X(s) \stackrel{d}{=} W(s)$$

Where W is a Gaussian white-noise, a generalized random field (i.e. that is only defined in the weak sense).
We know that the solution X is a Matérn GRF with regularity $\nu = \alpha - \frac{\text{dim}}{2}$.
We will note $\mathfrak{D} = \kappa^2 - \Delta$.

Modified Matérn SPDE

We will here look at a kind of simplified version of the Matérn SPDE, that is pointwise-defined:

$$\mathfrak{D}X \stackrel{d}{=} Y$$

Where Y is an Exponential random field, and the solution X is a Matérn GRF with $\nu = \frac{5}{2}$. This equation can be linked to the first one with the recurrence relation:

$$\mathfrak{D}X_\alpha \stackrel{d}{=} X_{\alpha-2} \Leftrightarrow X_\alpha \stackrel{d}{=} \mathfrak{D}^{-1}X_{\alpha-2}$$

such that $(\kappa^2 - \Delta)^\alpha X^\alpha(s) \stackrel{d}{=} W(s)$

KL expansion of an Exponential GRF

We want the KL expansion at rank N of a field Y with $C_Y(x_1, y_1) = \sigma_Y^2 \exp\left(-\frac{|x_1 - y_1|}{\eta}\right)$ on a segment $[-\frac{L}{2}, \frac{L}{2}]$ of size $L \in \mathbb{R}$:

$$\widetilde{Y}(\omega) = \sum_{n=1}^{N} \sqrt{\lambda_n}\, \phi_n(s)\, \xi_n(\omega)$$

KL expansion of an Exponential GRF

We know[1] that:

$$\lambda_n = \frac{2\eta\sigma_Y^2}{\eta^2 w_n^2 + 1},$$

and

$$\phi_n(x) = \frac{1}{\sqrt{(\eta^2 w_n^2 + 1)L/2 + \eta}}\left[\eta w_n \cos(w_n x) + \sin(w_n x)\right],$$

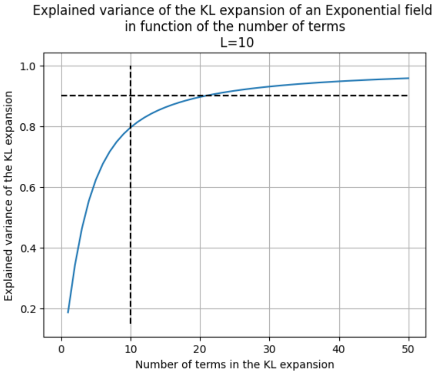
Where w_n are positive roots of the characteristic equation:

$$(\eta^2 w^2 - 1)\sin(wL) = 2\eta w \cos(wL).$$

[1] Based on the article *An efficient, high-order perturbation approach for flow in ran-dom porous media via Karhunen–Loeve and polynomial expansions* by Dongxiao Zhang and Zhiming Lu

KL expansion of an Exponential GRF

The sum of truncated eigenvalues correspond to the part explained variance of the expanded field.



Multivariate Polynomial Chaos

For the case where we want to expand a multivariate random variable $U(\omega)$, the sum is made on a family of multi-indexes that assemble in each term several Hermite polynomials evaluated at different independant Gaussian variables, making the expansion a multidimensional representation [2].
Let $\lambda(p)$ be the sub-set of multi-indices such that they correspond to a p^{th} - order set:

$$\lambda(p) = \left\{ \gamma : \sum_{i=1}^{N} \gamma_i = p \right\}$$

[1] From the book *Spectral Methods for Uncertainty Quantification*, O.P. Le Maître and O.M. Knio

Multivariate Polynomial Chaos

We can construct the set of p^{th} - order polynomials to represent our multidimensional random space, by assembling Hermite polynomials evaluated at different ξ_i as such:

$$\mathcal{H}(\boldsymbol{\xi}) := \left\{ \bigcup_{\gamma \in \lambda(p)} \prod_{i=1}^{N} \psi_{\gamma_i}(\xi_i) \right\}.$$

Multivariate Polynomial Chaos

Weighted tensor product between the vectors of Hermite polynomials evaluated at the different random variables $(H_0(\xi_1), H_1(\xi_1), ...)^T$, $(H_0(\xi_2), H_1(\xi_2), ...)^T$, ..., $(H_0(\xi_N), H_1(\xi_N), ...)^T$.
For example, in the 2D case ($\boldsymbol{\xi} = (\xi_1, \xi_2)$), the Hermite expansion can be expressed as:

$$\begin{aligned} U = {} & u_0\psi_0 + u_1\psi_1(\xi_1) + u_2\psi_1(\xi_2) \\ & + u_{11}\psi_2(\xi_1) + u_{21}\psi_1(\xi_2)\psi_1(\xi_1) + u_{22}\psi_2(\xi_2) \\ & + u_{111}\psi_3(\xi_1) + u_{211}\psi_1(\xi_2)\psi_2(\xi_1) + u_{221}\psi_2(\xi_2)\psi_1(\xi_1) \\ & + u_{222}\psi_3(\xi_2) + u_{1111}\psi_4(\xi_1) + \cdots \end{aligned}$$

The neural network

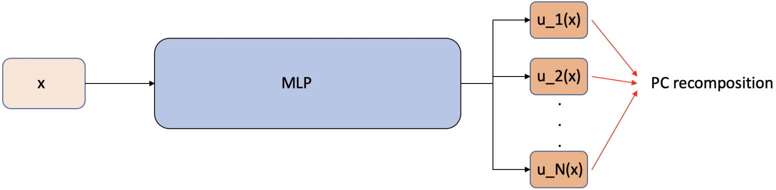


Figure: The idea of a PINN with PC-modes output

The Loss function

For the training of one neural network, we take a maximal order of Hermite polynomials for the PC expansion N_{PC}, and a maximal degree of KL expansion N_{KL}. The number of terms in the tensor product (and the number of weights) is:

$$\mathcal{N} := \binom{N_{PC}}{N_{KL} + N_{PC}}$$

Thus the output of our neural network will be:

$$U_\theta^{NN}(x) = \begin{pmatrix} u_0(x) \\ \vdots \\ u_{\mathcal{N}}(x) \end{pmatrix}$$

The Loss function

For $\mathfrak{D}X = Y$, we will compute the truncated KL expansion of Y:

$$Y_{KL}(\mathrm{x}) = \sum_{i=1}^{N_{KL}} \xi_i \sqrt{\lambda_i^Y} f_i^Y(\mathrm{x})$$

Using the same source of randomness, we have the PC reconstruction of the field generated by the NN:

$$PC[U_\theta^{NN}(x)] = \langle U_\theta^{NN}(x), \mathcal{H}(\boldsymbol{\xi}) \rangle$$

The Loss function

$$\mathcal{L}_\theta = \left\| \sum_{i=1}^{N_{KL}} \xi_i \sqrt{\lambda_i^Y} f_i^Y(\mathrm{x}) - \mathfrak{D}\left\langle \begin{pmatrix} u_0(\mathrm{x}) \\ \vdots \\ u_{\mathcal{N}}(\mathrm{x}) \end{pmatrix}, \ \mathcal{H}(\boldsymbol{\xi}) \right\rangle \right\|^2$$

This formula is meaned over K draws of gaussian vectors for one epoch (and one batch of collocation points).

Results

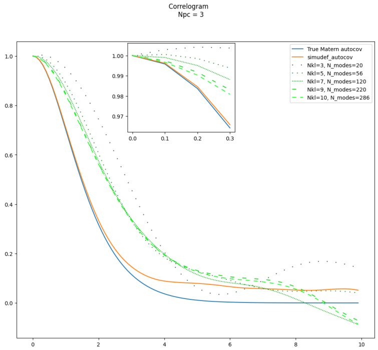
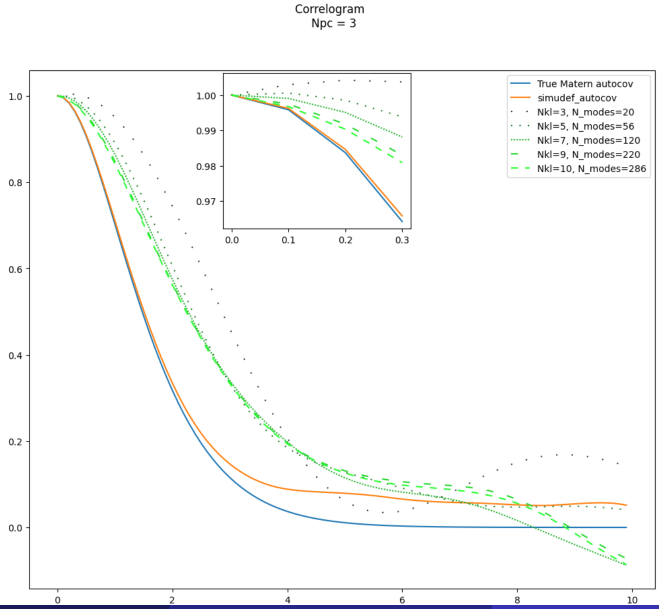


Figure: Trained neural networks results - Meaned correlograms of each model, for 500 samples.

Hyperparameters: learning rate = 10^{-3}, number of epochs = 5000, 8 neural layers of size 64, 1024 draws of gaussian vectors per epoch, and a collocation points batch size of 256 per epoch.

Results



Correlogram
Npc = 3
True Matern autocov
simudef_autocov
Nkl=3, N_modes=20
Nkl=5, N_modes=56
Nkl=7, N_modes=120
Nkl=9, N_modes=220
Nkl=10, N_modes=286
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
1.00
0.99
0.98
0.97
0.0
0.1
0.2
0.3

Results - error metric

We want to compare empirically our trained neural networks using the correlograms with more than our eyes. We simply use a squared mean distance between each correlogram, but we wanto compute this distance where the autocovariance values are significative.
Practical correlation range of our random fields:

$$\rho = \sqrt{8 * \nu * \frac{1}{\kappa}}$$

Results

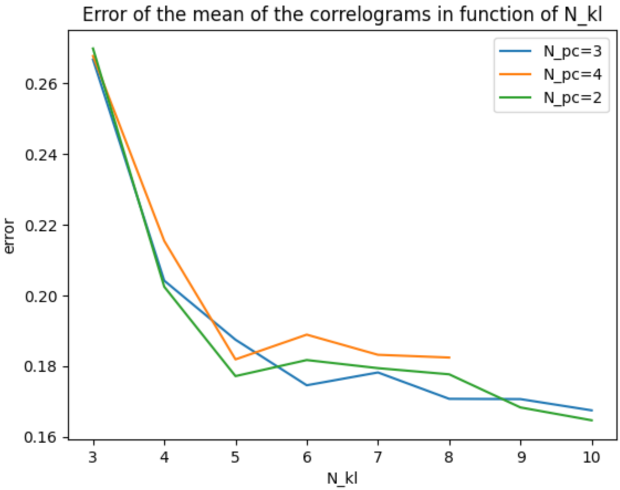


Figure: Error metrics

Libraries

- **JAX**
  - Python library for Deep Learning, function-oriented.
  - Provides tools for manual vectorization and just-in-time compilation.
  - Additional JAX-based libraries used:
    - Equinox
    - Optax
    - Jinns
- **chaospy**
  - Used for Hermite polynomial construction.
  - Incompatible with JAX, requiring experimental features.
  - Had to use CPU callback to handle integration.

$N_{KL} \downarrow N_{PC} \rightarrow$	3
5	$\mathcal{N} = 56,\ 513\ s$
6	$\mathcal{N} = 84,\ 794\ s$
7	$\mathcal{N} = 120,\ 2045\ s$
8	$\mathcal{N} = 165,\ 3894\ s$
9	$\mathcal{N} = 220,\ 5596\ s$
10	$\mathcal{N} = 286,\ 8534\ s$

Table: Some computation times, with the corresponding number of multivariate PC expansion modes $\mathcal{N}$ for trainings with hyperparameters: learning rate $= 10^{-3}$, number of epochs $= 5000$, 8 neural layers of size 64, 1024 draws of gaussian vectors per epoch, and a collocation points batch size of 256 per epoch.

For the same hyperparameters, with $N_{KL} = 9$ and $N_{PC} = 5$, we have $\mathcal{N} = 2002$ and a computation time of $61087s$

Conclusion

- Introduced stochastic PDEs and random field representations.
- Integrated PCE and KL expansions into PINNs.
- Open issues: training stability, variance representation.
- Prepared the terrain for Generative models based on probability distribution metrics and weak form of the problems.

References

Denis Allard (2012), *Statistiques spatiales: introduction à la géostatistique*.
R. J. Adler (1981), *The Geometry of Random Fields*.
Bertil Matérn (1960), *Spatial Variation*.
John B. Walsh (1986), *An Introduction to Stochastic Partial Differential Equations*.
Finn Lindgren, Håvard Rue, and Johan Lindström (2011), *An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach*.
Ivo Babuška, Fabio Nobile, and Raul Tempone (2007), *A stochastic collocation method for elliptic partial differential equations with random input data*.

References

Dongkun Zhang, Ling Guo, and George E. Karniadakis (2019), *Learning in Modal Space: Solving Time-Dependent Stochastic PDEs Using Physics-Informed Neural Networks*.

Dongxiao Zhang and Zhiming Lu (2003), *An efficient, high-order perturbation approach for flow in random porous media via Karhunen–Loeve and polynomial expansions*.

O. P. Le Maître and O. M. Knio (2010), *Spectral Methods for Uncertainty Quantification*.

Maziar Raissi, Paris Perdikaris, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*.