A robust solution of a statistical inverse problem in multiscale computational mechanics using an artificial neural network

Florent Pled, Christophe Desceliers, Tianyu Zhang

A robust solution of a statistical inverse problem in multiscale computational mechanics using an artificial neural network

Florent Pled[a,*], Christophe Desceliers[a], Tianyu Zhang[a]

[a] *Univ Gustave Eiffel, MSME UMR 8208, F-77454, Marne-la-Vallée, France*

Abstract

This work addresses the inverse identification of apparent elastic properties of random heterogeneous materials using machine learning based on artificial neural networks. The proposed neural network-based identification method requires the construction of a database from which an artificial neural network can be trained to learn the nonlinear relationship between the hyperparameters of a prior stochastic model of the random compliance field and some relevant quantities of interest of an *ad hoc* multiscale computational model. An initial database made up with input and target data is first generated from the computational model, from which a processed database is deduced by conditioning the input data with respect to the target data using the nonparametric statistics. Two- and three-layer feedforward artificial neural networks are then trained from each of the initial and processed databases to construct an algebraic representation of the nonlinear mapping between the hyperparameters (network outputs) and the quantities of interest (network inputs). The performances of the trained artificial neural networks are analyzed in terms of mean squared error, linear regression fit and probability distribution between network outputs and targets for both databases. An *ad hoc* probabilistic model of the input random vector is finally proposed in order to take into account uncertainties on the network input and to perform a robustness analysis of the network output with respect to the input uncertainties level. The capability of the proposed neural network-based identification method to efficiently solve the underlying statistical inverse problem is illustrated through two numerical examples developed within the framework of 2D plane stress linear elasticity, namely a first validation example on synthetic data obtained through computational simulations and a second application example on real experimental data obtained through a physical experiment monitored by digital image correlation on a real heterogeneous biological material (beef cortical bone).

Keywords: Machine learning, Uncertainty quantification, Probabilistic modeling, Statistical inverse problem, Random heterogeneous materials, Random multiscale modeling
2010 MSC: 62M45, 62M40, 65C05, 65C20, 74G75, 74S60, 74Q05, 62P10, 62P30

1. Introduction

1.1. Multiscale statistical inverse problem

The present paper concerns the mechanical characterization and identification of elastic properties for heterogeneous materials with a complex microstructure that may be considered as a random linear elastic medium. The high complexity level and multiphase nature of such microstructures do not allow for a proper description and modeling of the morphological and mechanical properties of their constituents at microscale. For such kind of materials, such as rock-like materials, concretes and cementitious materials, natural or synthetic composites and biological materials, a stochastic modeling of the apparent elastic properties of the

*Corresponding author
Email addresses: florent.pled@univ-eiffel.fr (Florent Pled), christophe.desceliers@univ-eiffel.fr (Christophe Desceliers), tianyu.zhang@univ-eiffel.fr (Tianyu Zhang)

microstructure can be constructed at a given mesoscale corresponding to the scale of the spatial correlation length of the microstructure. The uncertainties on the mechanical properties of such random heterogeneous materials are modeled by a non-Gaussian random elasticity (or compliance) field [1, 2] whose prior stochastic model is constructed within the framework of probability theory and information theory, that are among the most robust and well-established theories based on a solid mathematical background for several centuries. Such stochastic models of uncertainties are classically implemented into deterministic computational models yielding stochastic computational models that require parallel and high-performance computing (HPC) for propagating the uncertainties in high stochastic dimension. A major and still open challenge concerns the statistical inverse identification of stochastic models in using available data coming from either forward numerical simulations performed with computational models or experimental measurements obtained by means of physical tests. The statistical inverse problem under consideration consists in finding the values of the hyperparameters of a prior stochastic model of the random compliance field corresponding to highly probable values for some given observed quantities of interest of an *ad hoc* computational model. Such a statistical inverse problem has been formulated in [3, 4] as a multi-objective optimization problem and solved by using a global optimization algorithm (genetic algorithm) in [3] and a fixed-point iterative algorithm in [4], both requiring many calls to the computational model, which may be time consuming in practice especially for real-time applications such as in biomechanics. During the last decade, many identification methodologies and numerical developments have been proposed for addressing the problem related to the statistical inverse identification of stochastic models of the random elasticity (or compliance) field in low or high stochastic dimension at macroscale and/or mesoscale for complex microstructures modeled by random heterogeneous isotropic or anisotropic linear elastic media [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 3, 4]. The proposed identification methods require solving a statistical inverse problem classically formulated as a stochastic optimization problem, which may be computationally expensive even using modern computing hardware with powerful multicores processors and tricky to implement into commercial softwares dedicated to real-time identification or digital twin applications. In addition, the data required for performing the identification has to be stored in memory on the computing device and always accessible, which may be difficult to manage depending on the available memory storage capacity.

1.2. Improvements of the multiscale identification method and novelty of the paper

In the present work, we propose an appealing alternative for addressing the aforementioned drawbacks and solving the statistical inverse problem related to the identification of an *ad hoc* stochastic model of the random compliance field within the framework of 2D plane stress linear elasticity theory by using Machine Learning (ML) approaches based on Artificial Neural Networks (ANNs) [25, 26, 27]. ANNs are among the most widely used algorithms in supervised ML techniques to construct and train predictive models that map inputs to outputs for feature or pattern recognition/detection/selection/extraction, clustering, classification, compression/filtering, fitting/regression, identification and/or prediction/forecasting purposes. ML algorithms, such as ANNs, use advanced computational methods to learn information directly from data (without relying on any analytical or numerical model describing the input-output relationship). Since the training algorithms based on gradient computations for the design of ANNs are well adapted to parallelization, the training can be performed in parallel and distributed across multicores central processing units (CPUs), graphics processing units (GPUs), or even scaled up to clusters of computers and clouds with multiple CPUs and/or GPUs for a better computational efficiency. With the recent development in HPC and massively parallel processing systems, modern GPUs are more efficient than general-purpose CPUs for manipulating a huge amount of data due to their highly parallel structure. Consequently, they turn out to be particularly well adapted to machine learning for accelerating the network training process for very large datasets (often referred to as big data). Lastly, the use of ML algorithms has surged in popularity over the last years primarily due to their high accuracy, their short training time (thanks to the use of GPUs) and the storage, accessibility and processing of lots of data.

In the present paper, the statistical inverse problem is formulated as a function approximation problem and solved by using an ANN trained from a numerical database constructed from the computational model.

1.3. Methodology proposed in the paper

The proposed neural network-based identification method consists in the following steps.

1. A (deterministic) forward computational model is constructed and parameterized by the compliance field at mesoscale of the material. The quantities of interest that are computed by the forward computational model are gathered into the deterministic vector $\boldsymbol{q}$.

2. The uncertainty quantification on the values of $\boldsymbol{q}$ is carried out by modeling the quantities of interest as a random vector $\boldsymbol{Q}$. The random quantities of interest are defined as the outputs of the stochastic forward computational model that is constructed by introducing the prior stochastic model of the random compliance field. Let $\boldsymbol{h}$ be the vector of the hyperparameters of this prior stochastic model that has to be identified by the statistical inverse problem given an observation of $\boldsymbol{Q}$.

3. Since the value of $\boldsymbol{h}$ is uncertain, then the hyperparameters are modeled as a random vector $\boldsymbol{H}$.

4. For each of the N_d independent realizations $\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(N_d)}$ of $\boldsymbol{H}$, the stochastic forward computational model is used for computing one realization of $\boldsymbol{Q}$, yielding N_d independent realizations $\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(N_d)}$ of the quantities of interest (see Figure 1). An initial database is then obtained for which the i-th element is the vector $\boldsymbol{x}^{(i)} = (\boldsymbol{q}^{(i)}, \boldsymbol{h}^{(i)})$.

5. It should be noted that the mapping between $\boldsymbol{Q}$ and $\boldsymbol{H}$ is random by construction. As a consequence, the supervised training of an ANN with the initial database cannot be efficient since a trained ANN is a deterministic mapping between its inputs and outputs. This is the reason why the initial database is then processed in substituting $\boldsymbol{Q}$ by another network input random vector $\widetilde{\boldsymbol{Q}}$ such that the mapping between $\widetilde{\boldsymbol{Q}}$ and $\boldsymbol{H}$ is (almost) deterministic. It would then make it possible to efficiently train an artificial neural network. In this paper, it is proposed to obtain the processed database by conditioning the initial database. The N_d vectors $\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(N_d)}$ are then replaced by the N_d vectors $\tilde{\boldsymbol{q}}^{(1)}, \ldots, \tilde{\boldsymbol{q}}^{(N_d)}$, respectively (see Figure 2). Additional details on the construction of vectors $\tilde{\boldsymbol{q}}^{(1)}, \ldots, \tilde{\boldsymbol{q}}^{(N_d)}$ are given later in the paper. Such a data conditioning is performed by using classical kernel smoothing techniques in nonparametric statistics [28, 29, 30, 31, 32] for computing conditional mathematical expectations.

6. A multilayer ANN can then be designed to learn the nonlinear relationship between the hyperparameters (network outputs) $\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(N_d)}$ and the quantities of interest (network inputs) $\tilde{\boldsymbol{q}}^{(1)}, \ldots, \tilde{\boldsymbol{q}}^{(N_d)}$ and trained using the processed database in a potentially computationally expensive offline phase (preliminary learning phase). The (best) trained ANN can then be used to identify the value $\boldsymbol{h}^*$ of the output vector $\boldsymbol{h}$ of hyperparameters for a given observed input vector $\boldsymbol{q}^{\text{obs}}$ of quantities of interest in a computationally cheap online phase (real-time computing phase) (see Figure 3).

7. Finally, the robustness of the proposed identification method can be further assessed by considering the observed vector of quantities of interest as an input random vector for which the probabilistic model can be constructed by using the maximum entropy (MaxEnt) principle [33, 34, 35, 36, 37, 38, 39, 32], thus allowing the experimental errors on the observed quantities of interest (induced by the measurement noise and/or the variabilities in the experimental configuration) to be taken into account.

It should be pointed out that such an identification procedure can be performed directly with no call to the computational model (during the online computing phase), this latter being used only for the generation of the database required to design the multilayer ANN (during the offline learning phase). As a consequence, the proposed neural network-based identification strategy is computationally cheap, easy to implement and use.

1.4. Outline of the paper

The remainder of the paper is structured as follows. Section 2 presents the forward computational models, namely the High Fidelity Computational Mechanical Model and the Homogenization Computational Model, introduced within the framework of linear elasticity theory and used to compute relevant quantities of

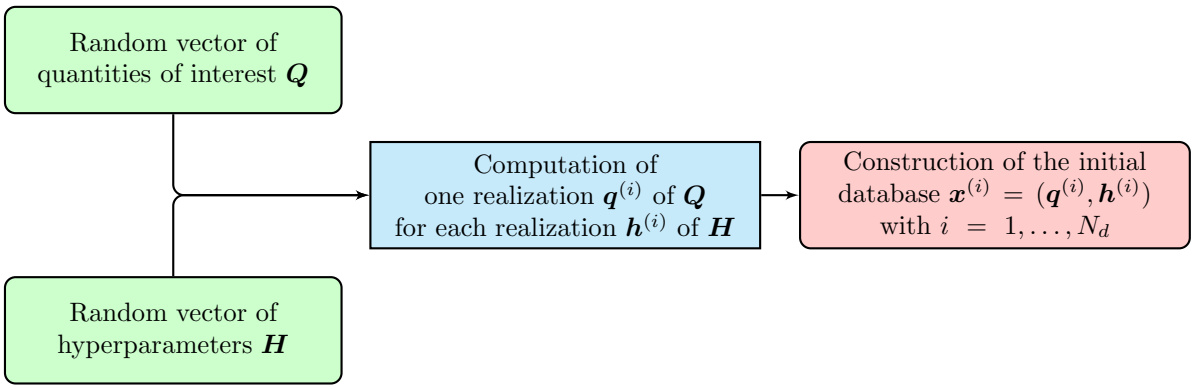


Figure 1: Flowchart for generating the initial database

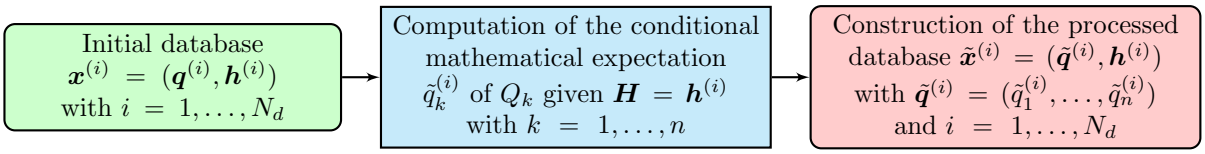


Figure 2: Flowchart for generating the processed database

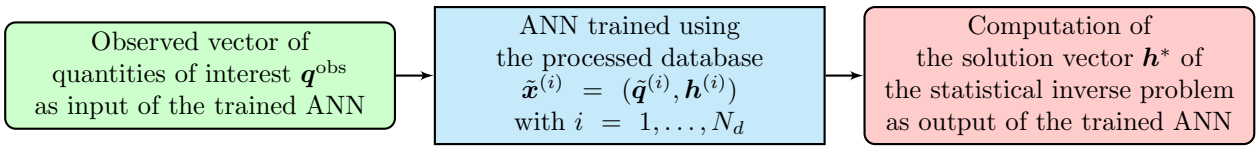


Figure 3: Flowchart for computing the solution of the statistical inverse problem

interest for the considered inverse identification problem to be solved. In Section 3, the prior stochastic model of the random compliance field that characterizes the apparent elastic properties of the random heterogeneous linear elastic medium under consideration is described and the associated hyperparameters to be identified are introduced. Section 4 is devoted to the construction of the initial database containing the network input and target data. Then, the statistical inverse problem is introduced and formulated as a function approximation problem in Section 5. Section 6 is devoted to the construction of the processed database obtained by conditioning the initial database and allowing for a robust identification of the solution of the statistical inverse problem. A statistical analysis of the initial and processed databases is then carried out in Section 7 for studying the sensitivity of the network target data with respect to the network input data. Section 8 deals with the design of the artificial neural network including the neural network architecture, the data partitioning and the training algorithm used to find the best predictive model. The performances of the multilayer neural networks trained with each of the initial and processed databases are then evaluated in terms of normalized mean squared error, linear regression fit and estimation of probability density function between network outputs and targets. An *ad hoc* probabilistic model for the input random vector of quantities of interest is presented in Section 9 in order to perform a robustness analysis of the network output with respect to the uncertainties on a given input. The capability of the proposed neural network-based identification method to efficiently solve the considered statistical inverse problem is shown though two numerical examples presented in Sections 10 and 11. The proposed approach implemented within the framework of 2D plane stress linear elasticity is first validated on synthetic data obtained by numerical simulations in Section 10 and then applied to real data obtained by means of experimental measurements on a real heterogeneous biological tissue (bovine cortical bone) in Section 11. Finally, Section 12 draws some

conclusions and suggests potentially beneficial directions for future research works.

2. Construction of a High Fidelity Computational Mechanical Model and a Homogenization Computational Model

Hereinafter, within the framework of linear elasticity theory, a High Fidelity Computational Mechanical Model (HFCMM) is constructed by using a classical displacement-based Finite Element Method (FEM) [40, 41] to compute a fine-scale displacement field of a heterogeneous elastic medium submitted to a given static external loading under the 2D plane stress assumption (see Figure 4). Such an assumption has been introduced only for better representing the experimental configuration for which experimental data are available for the numerical application presented in this paper. Consequently, the fine-scale vector-valued displacement field is only calculated on a 2D open bounded domain Ω^{macro} of $\mathbb{R}^2$ that is occupied by a heterogeneous linear elastic medium. In the following, we will consider a 2D square domain $\Omega^{\text{macro}} \subset \mathbb{R}^2$ defined in a fixed Cartesian frame (O, x_1, x_2) of $\mathbb{R}^2$ with macroscopic dimensions 1×1 cm^2. A given external line force field $\boldsymbol{f}$ is applied on the top part Γ_D^{macro} of the boundary $\partial\Omega^{\text{macro}}$ of Ω^{macro}, while the right and left parts are both stress-free boundaries and the bottom part $\Gamma_N^{\text{macro}} \subset \partial\Omega^{\text{macro}}$ is assumed to be fixed (see Figure 4). Without loss of generality, we assume no body force field within Ω^{macro}, the effects of gravity being neglected. Deterministic line force field $\boldsymbol{f}$ is uniformly distributed along the (downward vertical) $-x_2$ direction with an intensity of 5 kN such that $\|\boldsymbol{f}\| = 5$ kN/cm $= 5\times10^5$ N/m. The HFCMM is constructed by using the FEM for which the 2D domain Ω^{macro} is discretized with a fine structured mesh consisting of 4-nodes linear quadrangular elements with uniform element size $h = 10$ μm $= 10^{-5}$ m in each spatial direction. The finite element mesh of domain Ω^{macro} then contains $1001\times1001 = 1\,002\,001$ nodes and $1\,000\times1\,000 = 10^6$ elements, with $2\,000\,000$ unknown degrees of freedom. Within the framework of 2D plane stress linear elasticity theory, the elasticity properties of the heterogeneous linear elastic medium are characterized by a compliance field $[S^{\text{meso}}]$ with values in $\mathbb{M}_3^+(\mathbb{R})$, where $\mathbb{M}_3^+(\mathbb{R})$ denotes the set of all the definite-positive symmetric real (3×3) matrices. For identification purposes, the observed quantities of interest are obtained by postprocessing the kinematics fields that are calculated by the HFCMM on a subdomain $\Omega^{\text{meso}} \subset \Omega^{\text{macro}}$ for which the dimensions 1×1 mm^2 do not have to correspond to those of a Representative Volume Element (RVE) of the material since scale separation assumption is not required in all the following. Hence, a first quantity of interest calculated by the HFCMM consists in the spatial dispersion coefficient $\delta^{\boldsymbol{\varepsilon}}$ that quantifies the level of spatial fluctuations of the linearized strain field $\boldsymbol{\varepsilon}$ around its spatial average $\underline{\boldsymbol{\varepsilon}}$ over Ω^{meso} and that is defined by

$$\delta^{\boldsymbol{\varepsilon}} = \frac{1}{\|\underline{\boldsymbol{\varepsilon}}\|_F}\left(\frac{1}{|\Omega^{\text{meso}}|}\int_{\Omega^{\text{meso}}}\|\boldsymbol{\varepsilon}(\boldsymbol{x}) - \underline{\boldsymbol{\varepsilon}}\|_F^2\, d\boldsymbol{x}\right)^{1/2} \quad \text{with} \quad \underline{\boldsymbol{\varepsilon}} = \frac{1}{|\Omega^{\text{meso}}|}\int_{\Omega^{\text{meso}}}\boldsymbol{\varepsilon}(\boldsymbol{x})\, d\boldsymbol{x}, \tag{1}$$

where $|\Omega^{\text{meso}}|$ denotes the measure of domain Ω^{meso} and $\|\cdot\|_F$ denotes the Frobenius norm. The second and third quantities of interest are the two characteristic lengths $\ell_1^{\boldsymbol{\varepsilon}}$ and $\ell_2^{\boldsymbol{\varepsilon}}$ that characterize the spatial fluctuations of $\boldsymbol{\varepsilon}$ around its spatial average $\underline{\boldsymbol{\varepsilon}}$ along the two spatial directions x_1 and x_2, respectively, and naively computed on domain Ω^{meso} in using a usual signal processing method (such as the periodogram method, for instance) although $\ell_1^{\boldsymbol{\varepsilon}}$ and $\ell_2^{\boldsymbol{\varepsilon}}$ should be dependent of spatial position $\boldsymbol{x}$ because of the nature of the problem. The interested reader is referred to [42] for the numerical computation of $\ell_1^{\boldsymbol{\varepsilon}}$ and $\ell_2^{\boldsymbol{\varepsilon}}$. Computing the quantities of interest $\delta^{\boldsymbol{\varepsilon}}$, $\ell_1^{\boldsymbol{\varepsilon}}$ and $\ell_2^{\boldsymbol{\varepsilon}}$ by using the HFCMM for any given fine-scale matrix-valued compliance field $[S^{\text{meso}}]$ allows defining a nonlinear mapping $\boldsymbol{\mathcal{M}}^{\text{HFCMM}}$ defined from $\mathbb{M}_3^+(\mathbb{R})$ into $(\mathbb{R}^+)^3$ such that

$$(\delta^{\boldsymbol{\varepsilon}},\, \ell_1^{\boldsymbol{\varepsilon}},\, \ell_2^{\boldsymbol{\varepsilon}}) = \boldsymbol{\mathcal{M}}^{\text{HFCMM}}([S^{\text{meso}}]). \tag{2}$$

It should be noted that when the length scale of the heterogeneities is very small with respect to the dimensions of domain Ω^{macro}, then the dimension of such a computational model can be very high and the computational cost incurred by such HFCMM can become prohibitive in practical applications. In standard practice, the usual numerical approach then consists in computing the coarse-scale (macroscale) displacement field instead of the fine-scale (mesoscale) displacement field, for instance by calculating the

(3×3) effective compliance matrix $[S^{\text{eff}}]$ in 2D plane stress linear elasticity at a larger scale using an *ad hoc* computational homogenization method. Among the existing computational homogenization methods (see for instance [43, 44, 45, 46, 47] and the references therein), the static uniform boundary conditions (SUBC) homogenization approach, in which Neumann boundary conditions (homogeneous stresses) are applied on the whole boundary of Ω^{meso}, is preferred to the kinematic uniform boundary conditions (KUBC) method, in which Dirichlet boundary conditions (homogeneous strains) are applied on the whole boundary of Ω^{meso}. Nevertheless, such a coarse-scale displacement field approach avoids resorting to the HFCMM since the coarse-scale displacement field does not bring a sufficient level of granularity in the information for performing the inverse identification of the material properties at the finer scale. Previous research works (see [3, 48, 4]) have been carried out to avoid the use of a HFCMM but the identification methodology requires solving a challenging multi-objective optimization problem involving several disconnected boundary value problems at the fine scale set on domains for which the dimensions are not too large with respect to the characteristic size of the heterogeneities at microscale. A major drawback of this identification method is that a multi-objective optimization problem has to be solved for each experimental data, which severely limits its use in practical applications. Also, the computational cost of this multi-objective optimization problem is non negligible with the current available computer resources and remains high whatever the optimization algorithm considered such as the genetic algorithm used in [3, 48] or the fixed-point iterative algorithm introduced in [4] for a better computational efficiency. Consequently, such an approach cannot be used for real-time or digital twin applications for instance, and currently requires performing parallel and distributed computations across powerful multicores CPUs to preserve an affordable computational cost. This is the reason why, in the present paper, it is proposed to use Machine Learning (ML) approaches based on Artificial Neural Networks (ANNs) [25, 26, 27] that avoid solving such a computationally expensive optimization problem and that allow implementing a dedicated software on devices with general-purpose (regular) CPUs. For introducing the last quantity of interest, a Computational Homogenization Model that implements the SUBC homogenization approach is constructed in using a finite element mesh of Ω^{meso} made of $101\times101 = 10\,201$ nodes and $100\times100 = 10^4$ quadrangular elements. This Computational Homogenization Model is then used for computing the vector $\boldsymbol{\ell}^{\text{eff}} = (\log([L^{\text{eff}}]_{11}), [L^{\text{eff}}]_{12}, [L^{\text{eff}}]_{13}, \log([L^{\text{eff}}]_{22}), [L^{\text{eff}}]_{23}, \log([L^{\text{eff}}]_{33})) \in \mathbb{R}^6$ whose components are defined from the 6 components of the invertible upper triangular real matrix $[L^{\text{eff}}]$ (with strictly positive diagonal entries $[L^{\text{eff}}]_{11}$, $[L^{\text{eff}}]_{22}$ and $[L^{\text{eff}}]_{33}$) corresponding to the Cholesky factorization of the effective compliance matrix $[S^{\text{eff}}] \in \mathbb{M}_3^+(\mathbb{R})$, that is $[S^{\text{eff}}] = [L^{\text{eff}}]^T[L^{\text{eff}}]$, where the superscript ^{T} denotes the transpose operator. Hence, computing the vector-valued quantity of interest $\boldsymbol{\ell}^{\text{eff}}$ by using the Computational Homogenization Model for any fine-scale matrix-valued compliance field $[S^{\text{meso}}]$ allows defining a nonlinear mapping $\boldsymbol{\mathcal{M}}^{\text{EFF}}$ defined from $\mathbb{M}_3^+(\mathbb{R})$ into $\mathbb{R}^6$ such that

$$\boldsymbol{\ell}^{\text{eff}} = \boldsymbol{\mathcal{M}}^{\text{EFF}}([S^{\text{meso}}]). \tag{3}$$

Additional details can be found in [3, 48, 4] for the explicit construction of both nonlinear mappings $\boldsymbol{\mathcal{M}}^{\text{HFCMM}}$ and $\boldsymbol{\mathcal{M}}^{\text{EFF}}$.

3. Prior stochastic model of the uncertainties on the matrix-valued compliance field

In the present work, the material is assumed to be heterogeneous and anisotropic with a complex microstructure that cannot be properly described and numerically characterized from the morphological and mechanical properties of its micro-constituents. Matrix-valued compliance field $[S^{\text{meso}}]$ then represents the apparent elasticity properties of a random heterogeneous anisotropic material at a given mesoscale which corresponds to the fine scale that has been introduced in Section 2 for the HFCMM. Since the material is random, matrix-valued compliance field $[S^{\text{meso}}]$ is then considered as uncertain and modeled as a matrix-valued random compliance field $[\boldsymbol{S}^{\text{meso}}]$ indexed by $\mathbb{R}^2$ and restricted to bounded domain Ω^{macro}. A prior stochastic model of 2D matrix-valued random compliance field $[\boldsymbol{S}^{\text{meso}}]$ is constructed as a block matrix decomposition of a 3D matrix-valued random compliance field $[\boldsymbol{S}]$ in the ensemble SFE^+ of non-Gaussian second-order stationary almost surely (a.s.) positive-definite symmetric real matrix-valued random fields introduced in [1] (see [32] for an overview of the existing stochastic models and associated random generators for non-Gaussian random elasticity or compliance matrices or fields). This ensemble is adapted to the

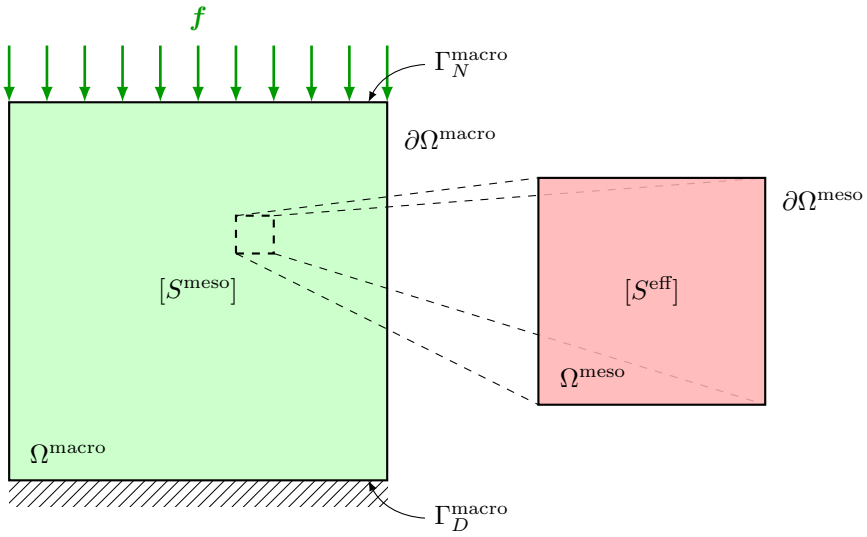


Figure 4: Linear elastic domains Ω^{macro} and Ω^{meso} for computing the fine-scale displacement fields by the High Fidelity Computational Mechanical Model ($\boldsymbol{\mathcal{M}}^{\text{HFCMM}}$) and the effective compliance matrix by the Computational Homogenization Model ($\boldsymbol{\mathcal{M}}^{\text{EFF}}$)

representation of elliptic stochastic partial differential operators (so as to ensure the existence and uniqueness of a second-order random solution to the underlying elliptic stochastic boundary value problem) and to the statistical inverse problem related to their experimental identification. Recall that, by construction, random compliance field $[\boldsymbol{S}^{\text{meso}}]$ satisfies a.s. the classical major and minor symmetry properties as well as the usual positive-definiteness properties and therefore takes its values in $\mathbb{M}_3^+(\mathbb{R})$. As a consequence of such a block decomposition for constructing random compliance field $[\boldsymbol{S}^{\text{meso}}]$, this latter is defined through a deterministic nonlinear mapping $\boldsymbol{\mathcal{G}}$ defined from $\mathbb{R}^6$ into $\mathbb{M}_3^+(\mathbb{R})$ as

$$[\boldsymbol{S}^{\text{meso}}] = \boldsymbol{\mathcal{G}}(\boldsymbol{U};\, \delta,\, \underline{\kappa},\, \underline{\mu}) \quad \text{with} \quad \boldsymbol{U} = \boldsymbol{\mathcal{U}}(\ell), \tag{4}$$

where δ is a positive bounded dispersion parameter such that $0 \leqslant \delta < \delta_{\text{sup}}$ with $\delta_{\text{sup}} = \sqrt{7/11} \approx 0.7977 < 1$ and controlling the level of statistical fluctuations exhibited by random compliance field $[\boldsymbol{S}^{\text{meso}}]$ around its mean function $[\underline{S}^{\text{meso}}]$, which is assumed to be independent of spatial position $\boldsymbol{x}$ and completely defined by a mean bulk modulus $\underline{\kappa}$ and a mean shear modulus $\underline{\mu}$ in the particular case of an isotropic mean elastic material, and where $\boldsymbol{\mathcal{U}}(\ell)$ is an explicit random algebraic or spectral representation of a second-order homogeneous normalized Gaussian $\mathbb{R}^6$-valued random field $\boldsymbol{U}$ indexed by $\mathbb{R}^2$ whose spatial correlation structure is parameterized by a unique spatial correlation length ℓ. The prior stochastic model of $[\boldsymbol{S}^{\text{meso}}]$ is finally parameterized by a four-dimensional vector-valued hyperparameter $\boldsymbol{h} = (\delta, \ell, \underline{\kappa}, \underline{\mu})$ belonging to the admissible set $\mathcal{H} = \mathcal{H}_1 \times \mathcal{H}_2 \times \mathcal{H}_3 \times \mathcal{H}_4 \subset (\mathbb{R}^+)^4$, with $\mathcal{H}_1 =]0\,, \delta_{\text{sup}}[$ and $\mathcal{H}_2 = \mathcal{H}_3 = \mathcal{H}_4 =]0\,, +\infty[$, and characterizing the complete probabilistic information of random compliance field $[\boldsymbol{S}^{\text{meso}}]$.

Additional details can be found in [1, 2, 32] for the fundamental (algebraic and statistical) properties of random compliance field $[\boldsymbol{S}]$, the explicit construction of the deterministic nonlinear mapping $\boldsymbol{\mathcal{G}}$ and an overview of the numerical methods allowing for the algebraic or spectral representation and numerical simulation (generation of realizations) of homogeneous Gaussian vector- or real-valued random fields. In the present work, we have used the spectral representation method (also called the Shinozuka method) initially introduced in [49, 50, 51] and later revisited and studied in [52, 53] from a mathematical standpoint, which is classical numerical simulation method based on the stochastic integral representation of homogeneous Gaussian random fields. The interested reader can refer to [42] for the algebraic representation of $[\boldsymbol{S}^{\text{meso}}]$ and the algorithm for generating independent realizations of $[\boldsymbol{S}^{\text{meso}}]$.

4. Construction of the initial database

In order to construct an *ad hoc* database for training an ANN that can be used for the statistical identification of the prior stochastic model of $[\boldsymbol{S}^{\text{meso}}]$, the unknown vector-valued hyperparameter $\boldsymbol{h} = (\delta, \ell, \underline{\kappa}, \underline{\mu})$ is modeled as a random vector $\boldsymbol{H} = (D, L, \underline{K}, \underline{M}) = (H_1, H_2, H_3, H_4)$ with statistically independent random components $H_1 = D$, $H_2 = L$, $H_3 = \underline{K}$ and $H_4 = \underline{M}$. Hence, mappings $\boldsymbol{\mathcal{M}}^{\text{HFCMM}}$, $\boldsymbol{\mathcal{M}}^{\text{EFF}}$ and $\boldsymbol{\mathcal{G}}$ respectively defined in (2), (3) and (4) allow for defining the random vector of quantities of interest $\boldsymbol{Q} = (Q_1, \ldots, Q_n)$ with values in $\mathbb{R}^n$ with $n = 9$, given random vector $\boldsymbol{H} = (H_1, \ldots, H_m)$ with values in $\mathcal{H} = \mathcal{H}_1 \times \ldots \times \mathcal{H}_m \subset \mathbb{R}^m$ with $m = 4$, such that

$$(Q_1,\, Q_2,\, Q_3) = \boldsymbol{\mathcal{M}}^{\text{HFCMM}}\Big(\boldsymbol{\mathcal{G}}(\boldsymbol{U};\, H_1,\, H_3,\, H_4)\Big) \quad \text{with} \quad \boldsymbol{U} = \boldsymbol{\mathcal{U}}(H_2), \tag{5a}$$

$$(Q_4,\, \ldots,\, Q_9) = \boldsymbol{\mathcal{M}}^{\text{EFF}}\Big(\boldsymbol{\mathcal{G}}(\boldsymbol{U};\, H_1,\, H_3,\, H_4)\Big) \qquad \text{with} \quad \boldsymbol{U} = \boldsymbol{\mathcal{U}}(H_2). \tag{5b}$$

The probabilistic model of random vector $\boldsymbol{H}$ is constructed by using the MaxEnt principle [33, 34, 35, 36, 37, 38, 39, 32] with the following algebraically independent constraints to be satisfied: (i) the components H_1, H_2, H_3 and H_4 of $\boldsymbol{H}$ are mutually statistically independent random variables, (ii) the support of the probability density function of $\boldsymbol{H}$ is a known bounded hypercube $\mathcal{H}_{\text{ad}} \subset \mathcal{H}$. Then, the MaxEnt principle leads to a uniform $\mathbb{R}^m$-valued random variable $\boldsymbol{H}$ with compact support $\mathcal{H}_{\text{ad}}$ and mutually statistically independent components. Note that in all the following, the reduced admissible set $\mathcal{H}_{\text{ad}} = [0.25\,, 0.65] \times [20\,, 250] \times [8.5\,, 17] \times [2.15\,, 5.00]$ in $[-] \times [\mu\text{m}] \times [\text{GPa}] \times [\text{GPa}]$ has been chosen sufficiently large so that the database can cover a large enough and realistic range of values of the hyperparameters for the application presented in Section 11 corresponding to a random heterogeneous microstructure made up of a biological tissue (bovine cortical bone) and by considering the results obtained in [48]. Furthermore, in practice, the bounds of admissible set $\mathcal{H}_{\text{ad}}$ may be *a posteriori* considered as incorrect if any component of output vector $\boldsymbol{h}^*$ is close to the corresponding bounds of $\mathcal{H}_{\text{ad}}$, which is not the case for the numerical examples presented in this paper.

The required numerical database should contain a set of network input and target (desired network output) vectors, where the input vectors define data regarding the random vector $\boldsymbol{Q}$ of quantities of interest, and the target vectors define data regarding the random vector $\boldsymbol{H}$ of hyperparameters. Such a database has been numerically simulated and constructed by using the random generator defined by (5). For each realization $\boldsymbol{h}^{(i)} = (h_1^{(i)}, \ldots, h_m^{(i)})$ of uniformly distributed random vector $\boldsymbol{H} = (H_1, \ldots, H_m)$, a realization of homogeneous normalized Gaussian random field $\boldsymbol{U}$ is generated using mapping $\boldsymbol{\mathcal{U}}$, then the corresponding realization of random compliance field $[\boldsymbol{S}^{\text{meso}}]$ is generated using mapping $\boldsymbol{\mathcal{G}}$, and finally the associated realization $\boldsymbol{q}^{(i)} = (q_1^{(i)}, \ldots, q_n^{(i)})$ of random vector $\boldsymbol{Q} = (Q_1, \ldots, Q_n)$ is numerically simulated using mappings $\boldsymbol{\mathcal{M}}^{\text{HFCMM}}$ and $\boldsymbol{\mathcal{M}}^{\text{EFF}}$. The construction of the database is then straightforward and it consists of N_d independent realizations $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N_d)}$ of random vector $\boldsymbol{X} = (\boldsymbol{Q}, \boldsymbol{H})$. Hence, each element of the database can be written as $\boldsymbol{x}^{(i)} = (\boldsymbol{q}^{(i)}, \boldsymbol{h}^{(i)})$ for $i = 1, \ldots, N_d$. Figure 5 provides a schematic representation of the key steps allowing the computation of the quantities of interest from the hyperparameters. Hereinafter, this database will be referred as the *initial database*.

5. Formulation of the statistical inverse problem

Solving the statistical inverse problem under consideration in this paper can be formulated as solving an optimization problem, as for instance calculating the value $\boldsymbol{h}^*$ of $\boldsymbol{H}$ as the Maximum A Posteriori (MAP) or the Maximum Likelihood Estimation (MLE). Another possible estimation of $\boldsymbol{h}^*$ can be chosen as the conditional mathematical expectation of $\boldsymbol{H}$ given $\boldsymbol{Q}$ is equal to a given observation $\boldsymbol{q}^{\text{obs}}$. Such estimations of $\boldsymbol{h}^*$ can be calculated by using usual nonparametric statistical methods [28, 29, 30, 31, 32] with the database constructed in Section 4. Nevertheless, these estimations of $\boldsymbol{h}^*$ require that the database is always available and sufficiently powerful CPUs for performing the computation in a reasonable computing time when digital twin applications are concerned, for instance. Since the required database may contain a large amount of data to be recorded, such a direct approach can be tricky to carry out in practice. An

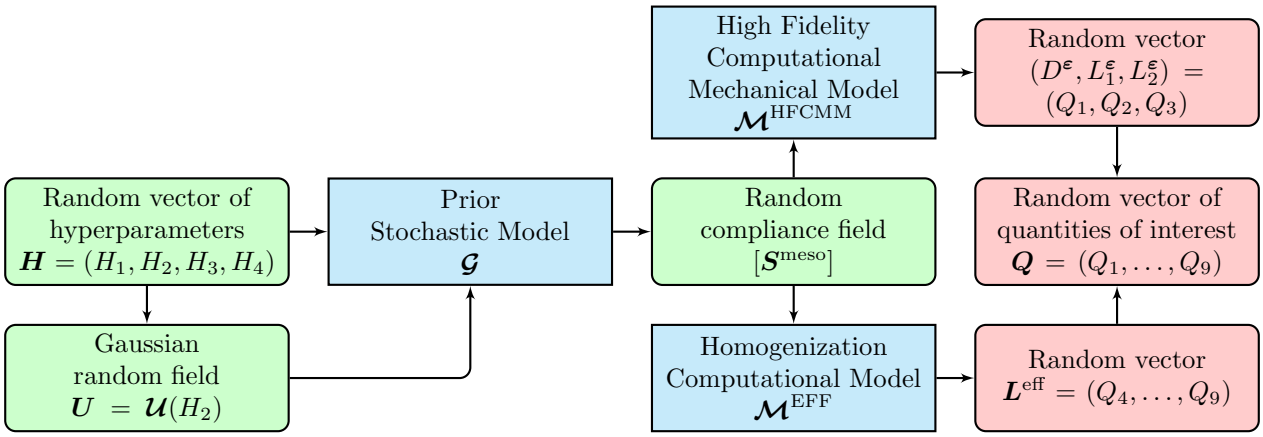


Figure 5: Flowchart for computing the quantities of interest from the hyperparameters. Blue blocks refer to the models, green blocks refer to the random input parameters and fields of the computational models and red blocks refer to the random output quantities of interest of the computational models.

alternative approach is proposed in the present work and consists in designing an ANN that can predict another probable value $\boldsymbol{h}^*$ of random vector $\boldsymbol{H}$ given $\boldsymbol{Q} = \boldsymbol{q}^{\text{obs}}$ with the available database for which the inputs will be the N_d independent realizations $\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(N_d)}$ of random vector $\boldsymbol{Q}$, and the corresponding targets will be the N_d independent realizations $\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(N_d)}$ of random vector $\boldsymbol{H}$. Indeed, ANNs are known for being particularly well-suited for addressing and solving function approximation and nonlinear regression problems. The statistical inverse problem related to the statistical identification of $\boldsymbol{h}^*$ can then be viewed as a function approximation problem and solved by using an ANN trained from the available database. The solution $\boldsymbol{h}^*$ of the statistical inverse problem can be simply defined as the output vector $\boldsymbol{h}^{\text{out}}$ of the trained ANN for the given input vector $\boldsymbol{q}^{\text{obs}}$. Within the framework of ML techniques based on ANNs, the *network input data* of the initial database will refer to the N_d independent realizations $\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(N_d)}$ of random vector $\boldsymbol{Q}$ and the *network target data* of the initial database will refer to the N_d independent realizations $\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(N_d)}$ of random vector $\boldsymbol{H}$. Nevertheless, it should be noted that since in (5a) and (5b), mapping $\boldsymbol{\mathcal{U}}$ is random for any input argument, then the mapping between $\boldsymbol{Q}$ and $\boldsymbol{H}$ is random too. As a consequence, the supervised training of an ANN with the initial database cannot be efficient since a trained ANN is a deterministic mapping between its inputs and outputs. This is the reason why $\boldsymbol{Q}$ is substituted by another network input vector $\widetilde{\boldsymbol{Q}}$ such that the mapping between $\widetilde{\boldsymbol{Q}}$ and $\boldsymbol{H}$ is (almost) deterministic. It would then make it possible to efficiently train an artificial neural network. In the next section, $\widetilde{Q}_k$ is defined as the conditional mathematical expectation of Q_k given $\boldsymbol{H}$. In practice, an observation $\tilde{q}_k^{\text{obs}}$ of $\widetilde{Q}_k$ is not available and cannot be deduced from a unique observation q_k^{obs} of Q_k since it would be equivalent to solve the statistical inverse problem. Nevertheless, we propose to calculate $\boldsymbol{h}^*$ as the output of the trained ANN with observation $\boldsymbol{q}^{\text{obs}} = (q_1^{\text{obs}}, \ldots, q_9^{\text{obs}})$ as input (instead of $\tilde{\boldsymbol{q}}^{\text{obs}} = (\tilde{q}_1^{\text{obs}}, \ldots, \tilde{q}_9^{\text{obs}})$).

6. Construction of the processed database by conditioning the initial database

For a robust computation of the solution $\boldsymbol{h}^*$ of the statistical inverse problem, the network input data consisting of the N_d inputs $q_k^{(1)}, \ldots, q_k^{(N_d)}$ and $\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(N_d)}$ for $k = 1, \ldots, n$ of the initial database are postprocessed and replaced with N_d new inputs $\tilde{q}_k^{(1)}, \ldots, \tilde{q}_k^{(N_d)}$ defined as the values taken by the conditional mathematical expectation $\mathbb{E}\{Q_k|\boldsymbol{H}\}$ of random variable Q_k given random vector $\boldsymbol{H}$ evaluated at $\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(N_d)}$, respectively. We then have

$$\tilde{q}_k^{(i)} = \mathbb{E}\{Q_k|\boldsymbol{H} = \boldsymbol{h}^{(i)}\} = \int_{\mathbb{R}} q\, p_{Q_k|\boldsymbol{H}}(q|\boldsymbol{h}^{(i)})\, dq, \tag{6}$$

where $q \mapsto p_{Q_k|\boldsymbol{H}}(q|\boldsymbol{h})$ is the conditional pdf of random variable Q_k given event $\boldsymbol{H} = \boldsymbol{h}$ for any $\boldsymbol{h} \in \mathcal{H}$. A nonparametric estimate of the conditional pdf $q \mapsto p_{Q_k|\boldsymbol{H}}(q|\boldsymbol{h})$ can be constructed by using the multivariate kernel density estimation method with a Gaussian kernel function, that is one of the most efficient and popular kernel smoothing techniques in nonparametric statistics [28, 29, 30, 31, 32], and the N_d independent realizations $q_k^{(1)}, \ldots, q_k^{(N_d)}$ and $\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(N_d)}$ of Q_k and $\boldsymbol{H}$, respectively. We then have

$$p_{Q_k|\boldsymbol{H}}(q|\boldsymbol{h}) = \frac{p_{Q_k,\boldsymbol{H}}(q,\boldsymbol{h})}{p_{\boldsymbol{H}}(\boldsymbol{h})} \simeq \frac{1}{b_{Q_k}} \frac{\displaystyle\sum_{i=1}^{N_d} \mathcal{K}\left(\frac{q - q_k^{(i)}}{b_{Q_k}}\right) \prod_{j=1}^{m} \mathcal{K}\left(\frac{h_j - h_j^{(i)}}{b_{H_j}}\right)}{\displaystyle\sum_{i=1}^{N_d} \prod_{j=1}^{m} \mathcal{K}\left(\frac{h_j - h_j^{(i)}}{b_{H_j}}\right)}, \tag{7}$$

where $(q, \boldsymbol{h}) \mapsto p_{Q_k,\boldsymbol{H}}(q,\boldsymbol{h})$ is the joint pdf of random vector $(Q_k, \boldsymbol{H})$ and $\boldsymbol{h} \mapsto p_{\boldsymbol{H}}(\boldsymbol{h})$ is the joint pdf of random vector $\boldsymbol{H}$, $x \mapsto \mathcal{K}(x)$ is a one-dimensional kernel function and the bandwidths b_{Q_k}, $b_{H_1}, \ldots, b_{H_m}$ are positive real values. In the present work, the Gaussian kernel function and the usual multidimensional optimal Silverman smoothing parameters computed using the so-called Silverman's rule of thumb [54] are chosen, and we then have

$$\mathcal{K}(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \quad \text{and} \quad b_S = \hat{\sigma}_S \left(\frac{4}{N_d(2+m+1)}\right)^{1/(4+m+1)},$$

where $\hat{\sigma}_S$ is a robust empirical estimate of the standard deviation of the real-valued random variable S, for $S = Q_k, H_1, \ldots, H_m$. Finally, the trapezoidal numerical integration method is employed to compute the integral of the one-dimensional function $q \mapsto q\, p_{Q_k|\boldsymbol{H}}(q|\boldsymbol{h}^{(i)})$ in (6). Note that for high-dimensional functions, the numerical integration could have been performed using a Markov Chain Monte Carlo (MCMC) method [55, 56, 57]. The conditioning of the initial database allows constructing a second database that consists of N_d elements $\tilde{\boldsymbol{x}}^{(1)}, \ldots, \tilde{\boldsymbol{x}}^{(N_d)}$ that are written as $\tilde{\boldsymbol{x}}^{(i)} = (\tilde{\boldsymbol{q}}^{(i)}, \boldsymbol{h}^{(i)})$ with $\tilde{\boldsymbol{q}}^{(i)} = (\tilde{q}_1^{(i)}, \ldots, \tilde{q}_n^{(i)})$ for $i = 1, \ldots, N_d$. Hereinafter, vectors $\tilde{\boldsymbol{q}}^{(1)}, \ldots, \tilde{\boldsymbol{q}}^{(N_d)}$ are modeled as statistically independent realizations of a vector-valued random variable $\widetilde{\boldsymbol{Q}}$ for which the probabilistic model is indirectly constructed by using the nonparametric statistics as presented in this section. In the following, the database containing the N_d elements $\tilde{\boldsymbol{x}}^{(1)}, \ldots, \tilde{\boldsymbol{x}}^{(N_d)}$ will be referred to as the *processed database*. Within the framework of ML techniques based on ANNs, the *network input data* of the processed database will refer to the N_d realizations $\tilde{\boldsymbol{q}}^{(1)}, \ldots, \tilde{\boldsymbol{q}}^{(N_d)}$ of random vector $\widetilde{\boldsymbol{Q}}$ and the *network target data* of the processed database will still refer to the N_d realizations $\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(N_d)}$ of random vector $\boldsymbol{H}$ as for the initial database.

In the present work, the complete initial (resp. processed) database consists of $N_d = 200\,000$ independent realizations of the 9-element random vector $\boldsymbol{Q}$ (resp. $\widetilde{\boldsymbol{Q}}$) and of the 4-element random vector $\boldsymbol{H}$. Such a large dataset (spanning the full range of the admissible output space $\mathcal{H}_{\text{ad}}$) is expected to cover the full range of the input space for which the ANN will be used after training. It should be mentioned that ANNs can reliably and accurately predict future outputs for new inputs belonging to the range for which they have been trained, but are generally not able to accurately extrapolate and generalize beyond (outside) this range. The values of the empirical estimate $\hat{\sigma}_S$ of the standard deviation for each input random variable $S = Q_1, \ldots, Q_9$ and each output random variable $S = H_1, H_2, H_3, H_4$ are reported in Table 1. In the following, both numerical databases (initial and processed) will be used to train a predictive model by designing an ANN that can reliably and accurately predict the output vector $\boldsymbol{h}^{\text{out}}$ for a given observed input vector $\boldsymbol{q}^{\text{obs}}$.

7. Statistical analysis of the initial and processed databases

A sensitivity analysis of the network target data with respect to the network input data has been performed for both the initial and processed databases. Figure 6 shows a classical estimate of the matrix of correlation coefficients between each of the components $Q_1, \ldots, Q_9$ (resp. $\widetilde{Q}_1, \ldots, \widetilde{Q}_9$) of random vector $\boldsymbol{Q}$ (resp. $\widetilde{\boldsymbol{Q}}$) and each of the components $H_1, \ldots, H_4$ of random vector $\boldsymbol{H}$ computed from the N_d network

Table 1: Empirical estimates $\hat{\sigma}_S$ of the standard deviation for each input random variable $S = Q_1, \ldots, Q_9$ and each output random variable $S = H_1, H_2, H_3, H_4$

Random variable S	Estimate $\hat{\sigma}_S$ of the standard deviation for S
Q_1	9.17×10^{-2}
Q_2	7.31×10^{-2}
Q_3	6.85×10^{-5}
Q_4	1.20×10^{-1}
Q_5	5.44×10^{3}
Q_6	1.84×10^{3}
Q_7	1.34×10^{-1}
Q_8	1.83×10^{3}
Q_9	1.47×10^{-1}
H_1	0.148
H_2	85.2 μm
H_3	3.14 GPa
H_4	1.05 GPa

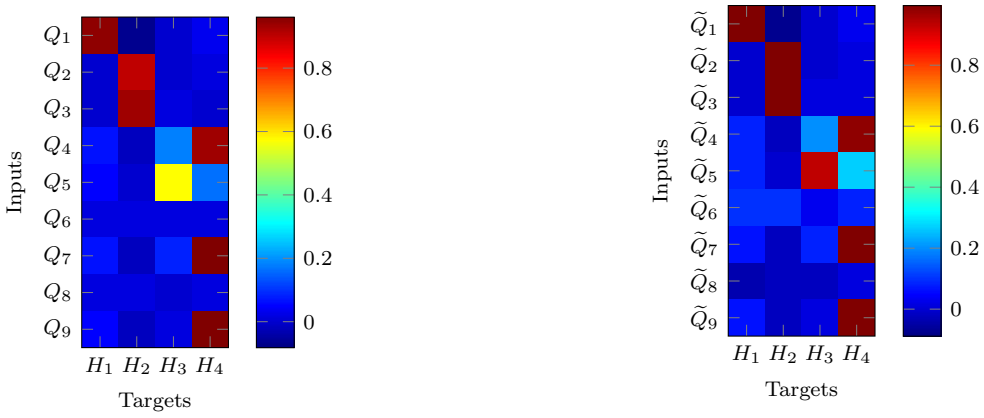
input vectors $\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(N_d)}$ (resp. $\tilde{\boldsymbol{q}}^{(1)}, \ldots, \tilde{\boldsymbol{q}}^{(N_d)}$) and corresponding target vectors $\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(N_d)}$ for the initial (resp. processed) database. First, we observe that random variable H_1 is highly correlated to random variable Q_1 (resp. $\widetilde{Q}_1$) and is almost not correlated with the other components of $\boldsymbol{Q}$ (resp. $\widetilde{\boldsymbol{Q}}$) for the initial (resp. processed) database. Secondly, random variable H_2 is strongly correlated to input random variables Q_2 and Q_3 (resp. $\widetilde{Q}_2$ and $\widetilde{Q}_3$) and have very small correlation with the other random components of $\boldsymbol{Q}$ (resp. $\widetilde{\boldsymbol{Q}}$) for the initial (resp. processed) database. Lastly, random variable H_3 is mostly correlated with random variable Q_5 (resp. $\widetilde{Q}_5$) and to a lesser extent with Q_4 (resp. $\widetilde{Q}_4$), while random variable H_4 is highly correlated with random variables Q_4, Q_7 and Q_9 (resp. $\widetilde{Q}_4$, $\widetilde{Q}_7$ and $\widetilde{Q}_9$) and to a lesser extent with Q_5 (resp. $\widetilde{Q}_5$) for the initial (resp. processed) database. It should be noted that the values of the highest correlation coefficients are higher for the processed database (containing the N_d independent realizations of random vector $\widetilde{\boldsymbol{Q}}$ as input data) than for the initial database (containing the N_d independent realizations of random vector $\boldsymbol{Q}$ as input data). Also, note that random variables Q_6 and Q_8 (resp. $\widetilde{Q}_6$ and $\widetilde{Q}_8$) are almost not correlated with any component of $\boldsymbol{H}$ for the initial (resp. processed) database, so that the corresponding realizations $q_6^{(1)}, \ldots, q_6^{(N_d)}$ and $q_8^{(1)}, \ldots, q_8^{(N_d)}$ (resp. $\tilde{q}_6^{(1)}, \ldots, \tilde{q}_6^{(N_d)}$ and $\tilde{q}_8^{(1)}, \ldots, \tilde{q}_8^{(N_d)}$) could have been removed from the initial (resp. processed) database.

8. Design of the Artificial Neural Network

In the present work, we focus on multilayer feedforward static neural networks (often referred to as series neural networks) that have only feedforward connections from the input layer (initial layer) to the first hidden layer, then from the first hidden layer to the second hidden layer and so on until the last hidden layer, and finally from the last hidden layer (penultimate layer) to the output layer (last layer). Recall that, while simple two-layer feedforward neural networks (with only one hidden layer and one output layer) have the ability to learn any multidimensional input-output relationship arbitrarily well given consistent data and enough hidden neurons in the hidden layer (see *e.g.* [58, 59, 60, 61, 62]), multilayer feedforward networks (with more than one hidden layer) are likely to learn complex input-output relationships more quickly and typically show better performance in some practical applications (see *e.g.* [63, 64] and the references therein).

8.1. Definition of the neural network architecture, data division and training algorithm

The architecture of the multilayer neural network involves a single input vector with 9 components and a single output vector with 4 components. The considered multilayer feedforward neural network is then



(a) Initial database with input random vector $\boldsymbol{Q} = (Q_1, \ldots, Q_9)$

(b) Processed database with input random vector $\widetilde{\boldsymbol{Q}} = (\widetilde{Q}_1, \ldots, \widetilde{Q}_9)$

Figure 6: Matrix of correlation coefficients between each of the components of random vector $\boldsymbol{Q}$ (resp. $\widetilde{\boldsymbol{Q}}$) and of random vector $\boldsymbol{H}$ estimated from the N_d network input vectors $\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(N_d)}$ (resp. $\tilde{\boldsymbol{q}}^{(1)}, \ldots, \tilde{\boldsymbol{q}}^{(N_d)}$) and corresponding target vectors $\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(N_d)}$ for the initial (resp. processed) database

composed of an input layer with 9 neurons, an output layer with 4 neurons, and one (or two) hidden layer(s) of neurons in between. Sigmoid hidden neurons, defined by a hyperbolic tangent sigmoid transfer function [65], are used in the hidden layer(s), while linear output neurons, defined by a linear transfer function, are used in the output layer. Various configurations have been tested for the two-layer (resp. three-layer) neural network with one (resp. two) hidden layer(s) and one output layer. For the two-layer neural network, the number of hidden neurons in the hidden layer is successively set to 4, 6, 8, 10, 15, 20, 25, 50, 75, 100, 150, 200, 250, 300, 350, 400, 450 and 500, for a total of 18 different configurations. For the three-layer neural network, the number of hidden neurons in each of the two hidden layers is successively set to 4, 6, 8, 10, 15, 20, 25, 50 and 75, for a total of 81 different configurations.

The input vectors and target vectors have been randomly divided into three distinct sets for training, validation and testing, with 70% of the complete dataset assigned to the training set, 15% to the validation set and 15% to the test set. Recall that the training and validation sets are used to build the model, while the test set is used to assess the performance of the trained model against test data that was set aside and not used during the training and validation processes in order to evaluate its ability to perform well on unseen data. More precisely, the training dataset is used to train the neural network with the backpropagation training algorithm [26] and adjust the network parameters, namely the weights and biases, according to the training performance function, that is the mean squared error of the training dataset. The validation set is used to measure network generalization and to prematurely interrupt training when generalization stops improving which is indicated by an increase in the validation performance function, that is the mean squared error of the validation dataset. Finally, the test set is used to provide an independent measure of network performance after training and validation. In the present data-fitting problem, the different training, validation and test sets have been simply defined by holding out an *ad hoc* percentage of the entire dataset. The training set consists of 70% of the complete dataset and therefore includes 140 000 samples of 13 elements (with 9 inputs and 4 outputs), while the validation and test sets are each set to 15% of the complete dataset and therefore include 30 000 samples of 13 elements (with 9 inputs and 4 outputs). The values of both the input and target vectors are preprocessed and mapped into the normalized range $[-1\,,1]$ before presenting to the neural network for training. After training, the network output vectors are then transformed back to the original scale (units) of the network target vectors. Such preprocessed and postprocessed transformations allow for the relative accuracy of the 4 components of output vectors to be optimized equally well although these 4 output elements have differing target value ranges.

The learning model has been constructed and developed from scratch without using transfer learning and a pretrained model and directly trained on the available training and validation datasets to fit the input

vectors and target vectors. The neural network has been set up with initial weight and bias values generated using the Nguyen-Widrow method [66] for each hidden layer and for the output layer. The neural network has been trained in batch mode, so that the weights and biases are adjusted and updated only once in each iteration corresponding to a full pass over the training dataset after all the input and target vectors in the training dataset are presented and applied to the network. Also, it has been retrained five times starting from several different initial conditions to ensure good network generalization and robust network performance and only the neural network with the best performance on the test dataset is considered for each configuration of the two-layer (resp. three-layer) neural network. As the network training may require considerable resources in terms of computational cost due to the large dataset size ($N_d = 200\,000$), we have used parallel and distributed GPU-based computing to speed up neural network training and simulation and manage large data by taking advantage of the massively parallelized architecture of GPUs. The neural network has been trained and simulated by using a high-performance GPU on a single computer with three GPUs and a hundred CPUs. The scaled conjugate gradient (SCG) algorithm has been chosen as training algorithm, since the conjugate gradient algorithms (in particular, the SCG algorithm) are among the most efficient algorithms for training large networks with thousands or millions of weights and biases on a broad class of problems, including function approximation and pattern recognition problems, with relatively small memory storage requirements compared to Jacobian-based training algorithms, such as the classical Levenberg-Marquardt (LM) and Bayesian Regularization (BR) algorithms. Also, note that the classical LM and BR algorithms have not been considered here, since these training algorithms are based on Jacobian computations that are not supported on GPU hardware (only gradient computations are supported on GPU devices). All the computations have been performed using the MATLAB Neural Network Toolbox™ [67] (now part of the Deep Learning Toolbox™) in conjunction with the Parallel Computing Toolbox™, the Statistics and Machine Learning Toolbox™ and the Optimization Toolbox™.

8.2. Analysis of the neural network performance after training

Once the neural network has fit the training dataset and generalized the input-output relationship using the validation dataset, it can be used to generate outputs for inputs it was not trained on and calculate its performance using the test dataset.

8.2.1. Measures of the neural network performance

The performances of the trained neural networks have been evaluated by (i) computing the normalized mean squared error between the network outputs and corresponding targets, (ii) performing a linear regression analysis, and (iii) displaying and comparing the marginal pdfs of each component of random vector $\boldsymbol{H} = (H_1, H_2, H_3, H_4)$ of hyperparameters estimated by using the univariate Gaussian kernel density estimation method [28] with the N_d network output data on the one hand and with the N_d network target data on the other hand.

The normalized mean squared error (mse) measures the neural network performance according to the mean of squared errors (corresponding to the average squared difference between outputs and targets) weighted by the squared distance between the maximum and minimum values for each target element, that is

$$\text{normalized mse} = \frac{1}{4}\sum_{j=1}^{4}\frac{1}{N_d}\sum_{i=1}^{N_d}\left(\frac{h_j^{\text{out},(i)} - h_j^{\text{target},(i)}}{h_j^{\text{target,max}} - h_j^{\text{target,min}}}\right)^2, \tag{8}$$

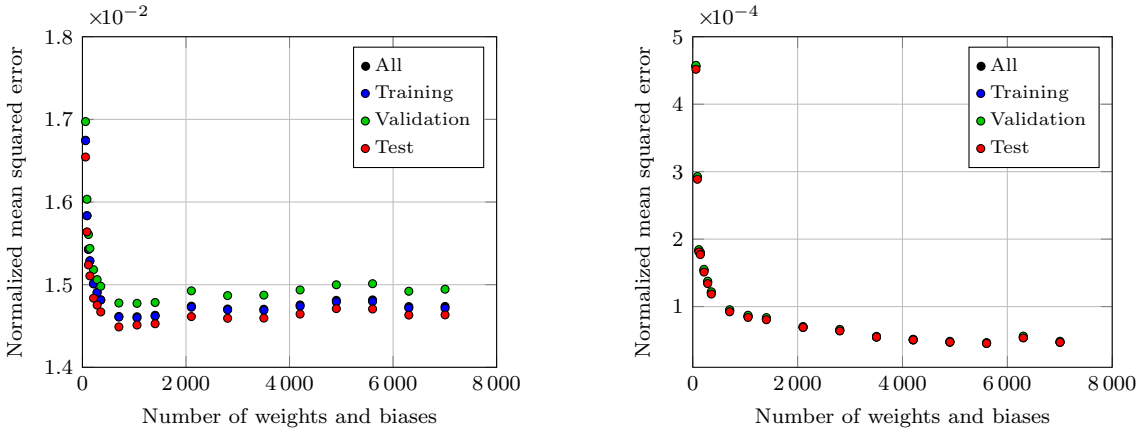
where $\boldsymbol{h}^{\text{out},(i)} = (h_1^{\text{out},(i)}, h_2^{\text{out},(i)}, h_3^{\text{out},(i)}, h_4^{\text{out},(i)})$ is the i-th network output vector, $\boldsymbol{h}^{\text{target},(i)} = (h_1^{\text{target},(i)}, h_2^{\text{target},(i)}, h_3^{\text{target},(i)}, h_4^{\text{target},(i)})$ is the corresponding target vector, $h_j^{\text{target,max}} = \max_{1\leqslant i\leqslant N_d} h_j^{\text{target},(i)}$ and $h_j^{\text{target,min}} = \min_{1\leqslant i\leqslant N_d} h_j^{\text{target},(i)}$ denote respectively the maximum and minimum values of the j-th target element.

The regression value (R-value) is defined and computed as the usual statistical estimate of the correlation coefficient between each output and the corresponding target, such that $R = 1$ (resp. R close to 1) indicates an exact (resp. almost) linear output-target relationship corresponding to a perfect (resp. very good) fit or correlation between output and target values, while $R = 0$ (resp. R close to 0) indicates a random (resp.

almost random) output-target relationship corresponding to no (resp. a very poor) fit or correlation. Since the neural network has multiple outputs with different ranges of values, the errors between outputs and corresponding targets have been normalized between -1 and 1 instead of their differing ranges so that the relative accuracy of each output element is optimized equally well (instead of optimizing and favoring the relative accuracy of the output elements with the largest range of values to the detriment of the output elements with the smallest range of values).

8.2.2. First measure: normalized mean squared error

As a first evaluation of the network performance, the normalized mse of the trained neural network is measured for the complete initial (resp. processed) dataset and for each of the training, validation and test subsets. The best trained neural network obtained using the CSG algorithm has been selected as the one with the best performance on the test set, *i.e.* the one that generalized best to the test set. Figure 7 shows the evolution of the normalized mse (plotted in a linear scale) with respect to the number of network parameters (weights and biases) for the two-layer neural network and for each of the initial and processed databases. For the initial database, the normalized mse slightly decreases and then reaches a plateau from a few hundreds of parameters with a relatively high value of about 1.5×10^{-2} for the complete, training, validation and test datasets, while for the processed database, the normalized mse sharply decreases with the number of parameters and then converges toward a very low value of 4×10^{-5} with several thousands of parameters. For the initial database, the best trained two-layer neural network contains 50 hidden neurons in the hidden layer with 704 parameters, while the best trained three-layer neural network contains 75 and 20 hidden neurons in the first and second hidden layers, respectively, with a total of 2 354 parameters. For the processed database, the best trained two-layer neural network contains 400 hidden neurons in the hidden layer with 5 604 parameters, while the best trained three-layer neural network contains 75 and 50 hidden neurons in the first and second hidden layers, respectively, with a total of 4 754 parameters. Figures 8 and 9 show graphical diagrams of the best trained two- and three-layer neural networks obtained for each of the initial and processed databases.

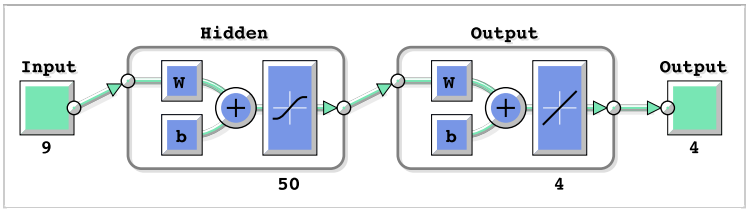


(a) Initial database

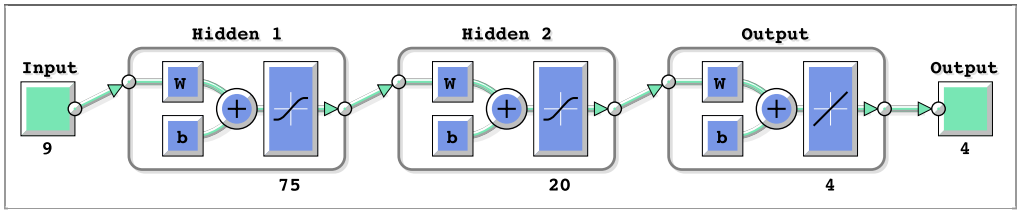
(b) Processed database

Figure 7: Evolution of the performance function (normalized mean squared error) with respect to the number of weights and biases of the two-layer neural network on the complete dataset (black symbols), training dataset (blue symbols), validation dataset (green symbols) and test dataset (red symbols) for (a) the initial database and (b) the processed database

Figures 10 and 11 show the graphs of the performance function (normalized mse) with respect to the number of training iterations for evaluating the training, validation and test performances of the best two- and three-layer trained neural networks (on the training, validation and test datasets, respectively) for each of the initial and processed databases. The normalized mse is plotted in a logarithmic scale. The network training continued until the validation performance function (normalized mean squared error on the validation dataset) failed to decrease after 6 iterations (validation checks). For the initial (resp. processed)

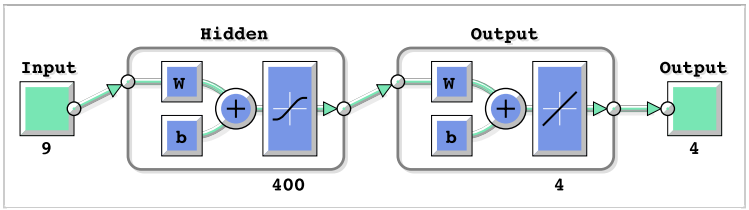


(a) Two-layer neural network with one hidden layer containing 50 tan-sigmoid hidden neurons and one output layer containing 4 linear output neurons

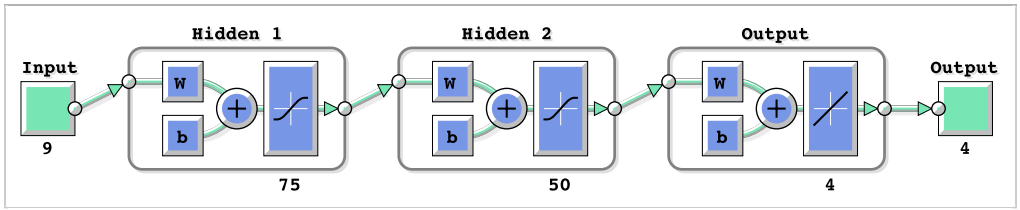


(b) Three-layer neural network with two hidden layers containing 75 and 20 tan-sigmoid hidden neurons, respectively, and one output layer containing 4 linear output neurons

Figure 8: Initial database: graphical diagrams of (a) the best two-layer feedforward neural network and (b) the best three-layer feedforward neural network



(a) Two-layer neural network with one hidden layer containing 400 tan-sigmoid hidden neurons and one output layer containing 4 linear output neurons

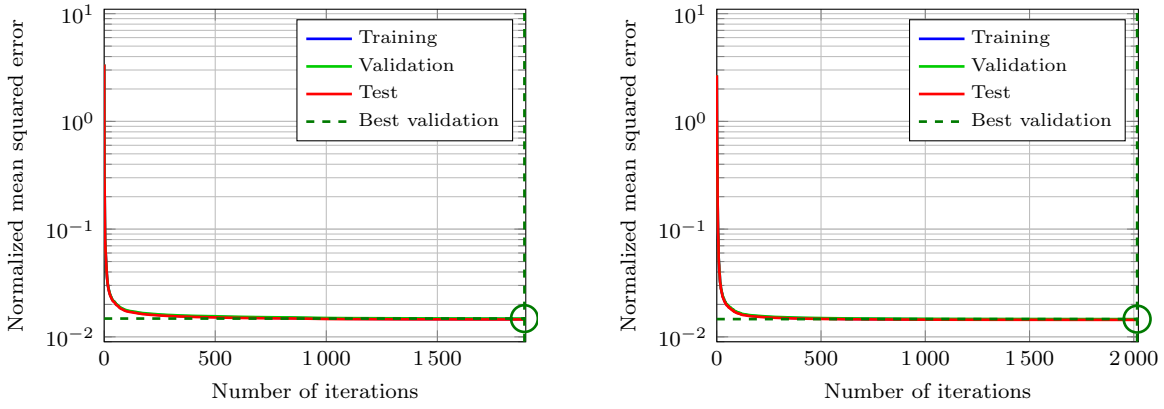


(b) Three-layer neural network with two hidden layers containing 75 and 50 tan-sigmoid hidden neurons, respectively, and one output layer containing 4 linear output neurons

Figure 9: Processed database: graphical diagrams of (a) the best two-layer feedforward neural network and (b) the best three-layer feedforward neural network

database, the validation performance function reached a minimum at iterations 1 892 and 2 022 (resp. 9 233 and 12 423) for the best two-layer and three-layer neural networks and the training continued for 6 more iterations before it stopped. The normalized mean squared errors rapidly decrease during the first iterations and then slowly converge until validation stops for each of the training, validation and test datasets. The performance curves (normalized mse versus number of iterations) are similar for both validation and test datasets, indicating no significant overfitting occurred. The normalized mean squared errors obtained at final iteration, where the best validation performance occurs, are given in Table 2 for the training, validation, test and complete datasets and for the initial and processed databases. For each of the two numerical databases and each of the two multilayer neural networks, the network performances are similar for each of the training, validation, test and complete datasets. However, the network performances obtained with the processed database, which are around 10^{-5}, are significantly better than that obtained with the initial

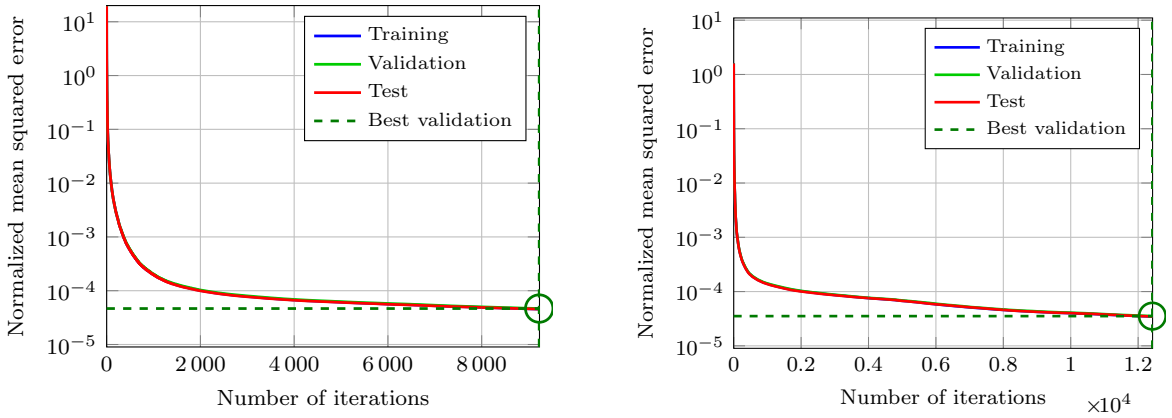
database, which are around 10^{-2}. Also, for each database, the best three-layer neural network shows slightly higher performances than the best two-layer neural network. Finally, the best overall network performance (normalized mse computed on the complete dataset) is obtained with the processed database and the three-layer neural network and is equal to $3.48{\times}10^{-5}$, and the training, validation and test performances are equal to 3.47×10^{-5}, 3.53×10^{-5} and 3.48×10^{-5}, respectively.



(a) Two-layer neural network

(b) Three-layer neural network

Figure 10: Initial database: evolution of the performance function (normalized mean squared error) with respect to the number of training iterations for the training (blue curve), validation (green curve) and test (red curve) datasets, with the best validation performance indicated by green dashed lines, for (a) the best two-layer neural network and (b) the best three-layer neural network



(a) Two-layer neural network

(b) Three-layer neural network

Figure 11: Processed database: evolution of the performance function (normalized mean squared error) with respect to the number of training iterations for the training (blue curve), validation (green curve) and test (red curve) datasets, with the best validation performance indicated by green dashed lines, for (a) the best two-layer neural network and (b) the best three-layer neural network

8.2.3. Second measure: linear regression between network outputs and targets

As a second evaluation of the network performance, the linear regression of the network outputs relative to targets is plotted across the complete dataset in Figures 12 and 13 for the initial and processed databases, respectively, and for the best three-layer neural network. Note that very similar regression plots have been obtained for the best two-layer neural network and are not reported here for the sake of brevity. For both numerical databases (initial and processed), the trained network output vectors have been computed for

Table 2: Normalized mean squared errors obtained for the best two-layer and three-layer neural networks trained from the initial and processed databases

	Initial database		Processed database	
Dataset	Two-layer neural network	Three-layer neural network	Two-layer neural network	Three-layer neural network
Training	1.46×10^{-2}	1.44×10^{-2}	4.55×10^{-5}	3.47×10^{-5}
Validation	1.48×10^{-2}	1.46×10^{-2}	4.66×10^{-5}	3.53×10^{-5}
Test	1.45×10^{-2}	1.44×10^{-2}	4.55×10^{-5}	3.48×10^{-5}
All	1.46×10^{-2}	1.45×10^{-2}	4.57×10^{-5}	3.48×10^{-5}

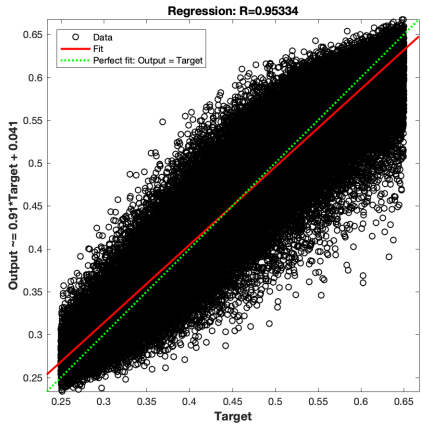
all the input vectors in the complete dataset, then the output and target vectors belonging to each of the training, validation and test subsets have been extracted, and finally the network outputs have been plotted with respect to targets for each of the training, validation and test subsets as well as for the complete dataset. Very similar trends have been observed for the complete dataset and for each of the training, validation, test data subsets separately, so that only the results for the complete dataset are displayed in Figures 12 and 13 for the sake of simplicity and conciseness. On the one hand, for the initial database, the best linear fit between network outputs and corresponding targets, although not perfect, is fairly good for the complete dataset (and for each data subset) with regression values (R-values) over 0.95 for dispersion parameter H_1, 0.96 for spatial correlation length H_2, 0.70 for mean bulk modulus H_3, and 0.98 for mean shear modulus H_4. Nevertheless, the scatter plots of the network outputs and corresponding targets are highly dispersed and show that some data points in the dataset have poor fits, especially for H_4. Such a large dispersion is due to the stochastic nature of the non-linear mapping defined in (5) between random vector $\boldsymbol{Q}$ of quantities of interest and random vector $\boldsymbol{H}$ of hyperparameters. On the other hand, for the processed database, the best linear fit between network outputs and corresponding targets is almost perfect for the complete dataset (and for each data subset) with regression values (R-values) very close to 1 for all components H_1, H_2, H_3 and H_4 of $\boldsymbol{H}$ (and for all data subsets). The network outputs track the targets with very small dispersion for each network output H_1, H_2, H_3 and H_4, showing a significantly better fit for the processed database than for the initial database.

8.2.4. Third measure: marginal probability density functions of the components of the output random vector

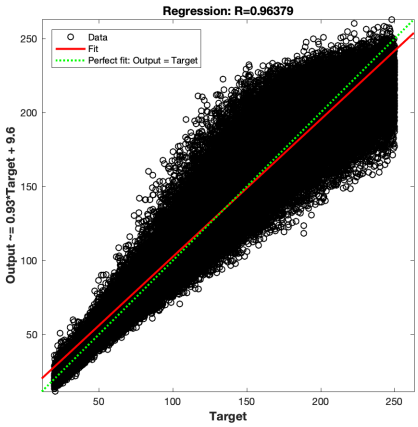
As a third evaluation of the network performance, the marginal pdfs p_{H_1}, p_{H_2}, p_{H_3} and p_{H_4} of each component of random vector $\boldsymbol{H} = (H_1, H_2, H_3, H_4)$ of hyperparameters, which are assumed to be uniform random variables, are estimated by using the univariate Gaussian kernel density estimation method [28] with the N_d network output data obtained from the initial (resp. processed) database with the best three-layer neural network, and compared to the uniform target pdfs and to the target pdfs estimated by using the univariate Gaussian kernel density estimation method with the N_d associated target data in Figure 14. The output pdfs constructed from the output vectors of the best neural network trained with the processed database perfectly match the associated target pdfs, while the output pdfs constructed from the output vectors of the best neural network trained with the initial database have a worse fit, especially for H_3.

8.2.5. Discussion

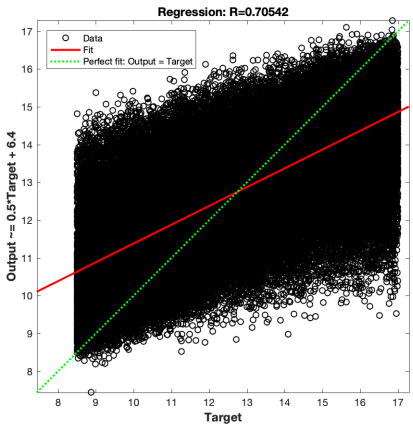
According to the aforementioned results concerning the network performances, the neural network trained with the processed database, that is to say obtained by conditioning the network input vectors contained in the initial database with respect to the network target vectors, can directly be used for identifying the value $\boldsymbol{h}^{\text{out}}$ of random hyperparameters $\boldsymbol{H}$ corresponding to a given observed vector $\boldsymbol{q}^{\text{obs}}$ of quantities of interest. The conditioning of the initial database then appears to be a determining key factor in obtaining an efficient trained neural network. For a given input vector $\boldsymbol{q}^{\text{obs}}$, the output vector $\boldsymbol{h}^{\text{out}}$ computed by the best neural network trained with the processed database corresponds to the solution $\boldsymbol{h}^{*}$ of the statistical inverse problem formulated in Section 5. Finally, computing network output vector $\boldsymbol{h}^{\text{out}}$ for any network



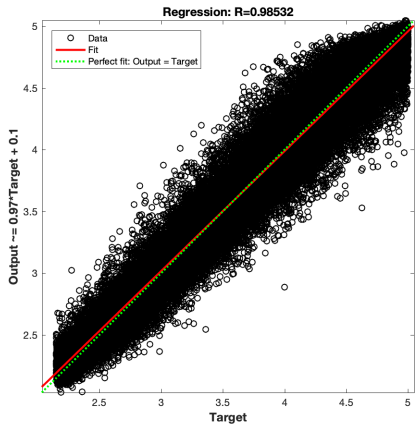
(a) dispersion parameter H_1



(b) spatial correlation length H_2 [μm]



(c) mean bulk modulus H_3 [GPa]



(d) mean shear modulus H_4 [GPa]

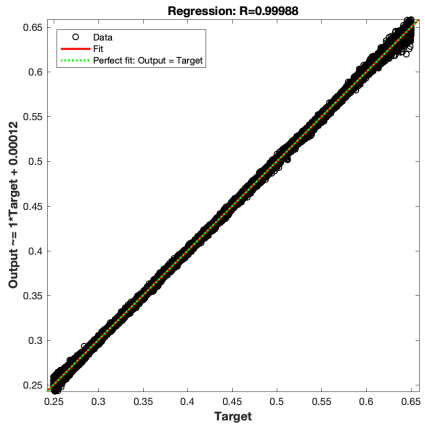
Figure 12: Initial database and three-layer neural network: linear regression between network outputs and corresponding targets for each random hyperparameter H_1, H_2, H_3 and H_4, for the complete dataset. In each plot, the network outputs and targets are represented by open black circles, the perfect fit (outputs exactly equal to targets) is represented by a dashed green line, and the best linear fit (linear regression between outputs and targets) is represented by a solid red line for the complete dataset. The regression value (R-value) is given at the top of each regression plot

input vector $\boldsymbol{q}^{\mathrm{obs}}$ allows for defining a deterministic nonlinear mapping $\boldsymbol{\mathcal{N}}$ defined from $\mathbb{R}^n$ into $\mathbb{R}^m$ as

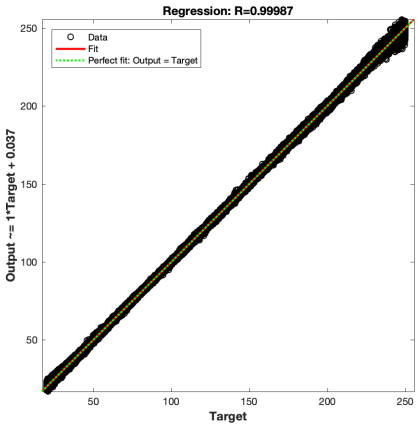
$$\boldsymbol{h}^{\mathrm{out}} = \boldsymbol{\mathcal{N}}(\boldsymbol{q}^{\mathrm{obs}}). \tag{9}$$

9. Robust solution of the statistical inverse problem

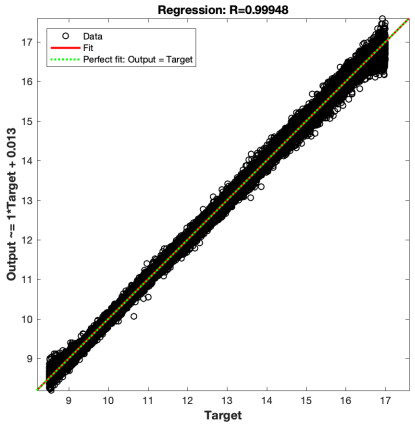
In order to assess the robustness of the proposed identification method by taking into account experimental errors (measurement errors and epistemic uncertainties) on the input vector $\boldsymbol{q}^{\mathrm{obs}} = (\delta^{\boldsymbol{\varepsilon}}_{\mathrm{obs}}, \ell^{\boldsymbol{\varepsilon}}_{\mathrm{obs},1}, \ell^{\boldsymbol{\varepsilon}}_{\mathrm{obs},2}, \boldsymbol{\ell}^{\mathrm{eff}}_{\mathrm{obs}})$ of quantities of interest, this latter can be considered and modeled as a random vector $\boldsymbol{Q}^{\mathrm{obs}} = (D^{\boldsymbol{\varepsilon}}_{\mathrm{obs}}, L^{\boldsymbol{\varepsilon}}_{\mathrm{obs},1}, L^{\boldsymbol{\varepsilon}}_{\mathrm{obs},2}, \boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}})$ where $\boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}} = (\log([\boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}}]_{11}), [\boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}}]_{12}, [\boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}}]_{13}, \log([\boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}}]_{22}), [\boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}}]_{23}, \log([\boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}}]_{33}))$ is the random vector with values in $\mathbb{R}^6$ whose components are deduced from the 6 components of the invertible upper triangular random matrix $[\boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}}]$ with values in $\mathbb{M}^+_3(\mathbb{R})$ (with positive-valued diagonal entries $[\boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}}]_{11}$, $[\boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}}]_{22}$ and $[\boldsymbol{L}^{\mathrm{eff}}_{\mathrm{obs}}]_{33}$) resulting from the Cholesky factorization of the random compliance matrix $[\boldsymbol{S}^{\mathrm{eff}}_{\mathrm{obs}}]$ with values in $\mathbb{M}^+_3(\mathbb{R})$, that



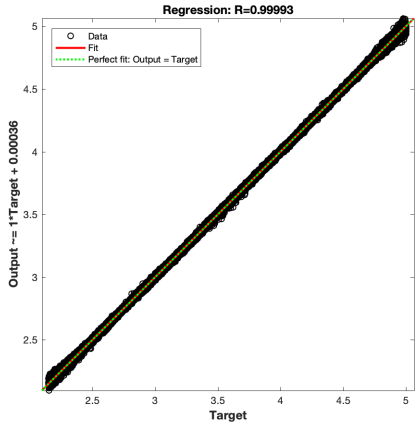
(a) dispersion parameter H_1



(b) spatial correlation length H_2 [μm]



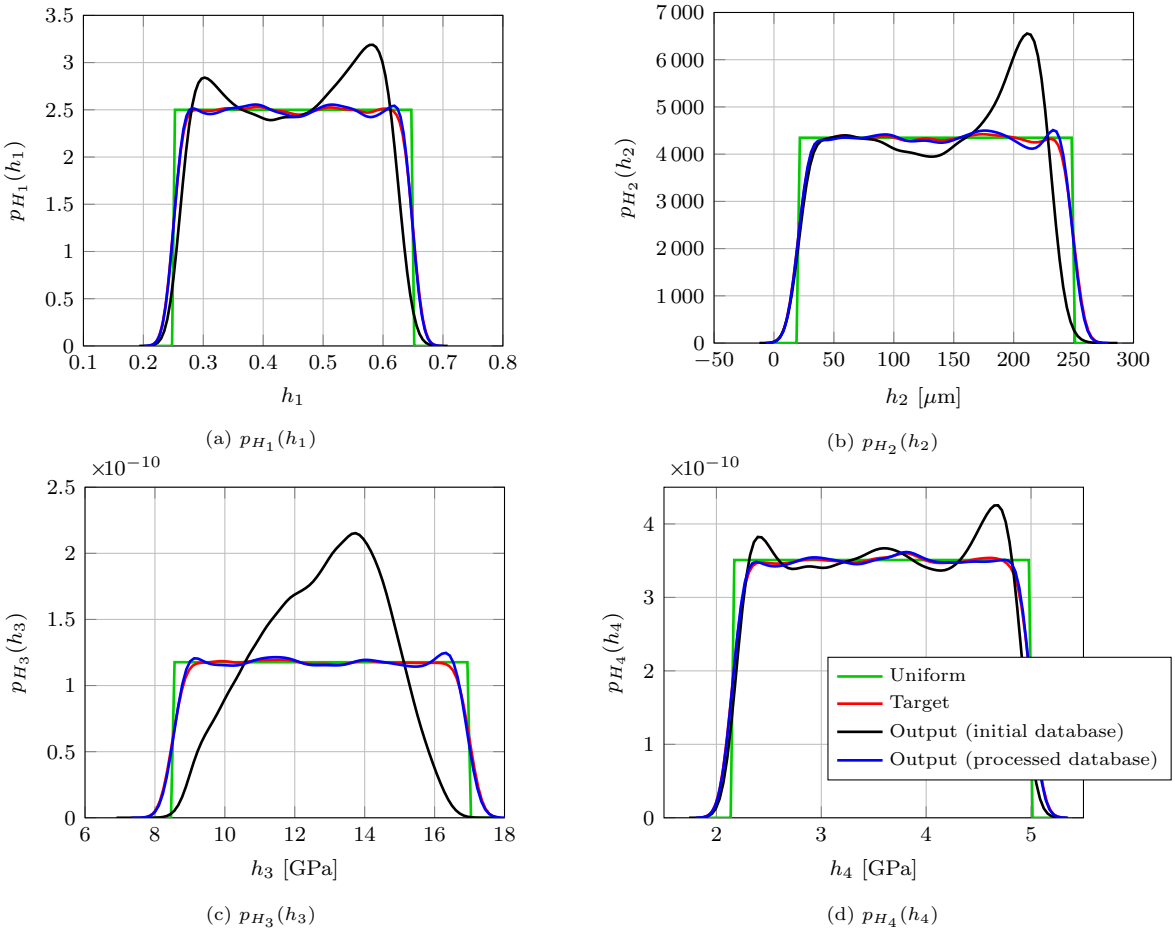
(c) mean bulk modulus H_3 [GPa]



(d) mean shear modulus H_4 [GPa]

Figure 13: Processed database and three-layer neural network: linear regression between network outputs and corresponding targets for each random hyperparameter H_1, H_2, H_3 and H_4, for the complete dataset. In each plot, the network outputs and targets are represented by open black circles, the perfect fit (outputs exactly equal to targets) is represented by a dashed green line, and the best linear fit (linear regression between outputs and targets) is represented by a solid red line for the complete dataset. The regression value (R-value) is given at the top of each regression plot

is $[\boldsymbol{S}^{\text{eff}}_{\text{obs}}] = [\boldsymbol{L}^{\text{eff}}_{\text{obs}}]^T[\boldsymbol{L}^{\text{eff}}_{\text{obs}}]$. The prior probabilistic model of $\boldsymbol{Q}^{\text{obs}}$ is constructed by having recourse to the MaxEnt principle [33, 34, 35, 36, 37, 38, 39, 32] based on the following algebraically independent constraints to be satisfied: (i) $D^{\boldsymbol{\varepsilon}}_{\text{obs}}$, $L^{\boldsymbol{\varepsilon}}_{\text{obs},1}$, $L^{\boldsymbol{\varepsilon}}_{\text{obs},2}$ and $\boldsymbol{L}^{\text{eff}}_{\text{obs}}$ are mutually statistically independent random variables, (ii) $D^{\boldsymbol{\varepsilon}}_{\text{obs}}$, $L^{\boldsymbol{\varepsilon}}_{\text{obs},1}$ and $L^{\boldsymbol{\varepsilon}}_{\text{obs},2}$ are a.s. $\mathbb{R}^+$-valued random variables for which the values are unlikely close to zero and consequently $\boldsymbol{E}\{\log(D^{\boldsymbol{\varepsilon}}_{\text{obs}})\}$, $\boldsymbol{E}\{\log(L^{\boldsymbol{\varepsilon}}_{\text{obs},1})\}$ and $\boldsymbol{E}\{\log(L^{\boldsymbol{\varepsilon}}_{\text{obs},2})\}$ are finite, (iii) the mean values $\mathbb{E}\{D^{\boldsymbol{\varepsilon}}_{\text{obs}}\}$, $\mathbb{E}\{L^{\boldsymbol{\varepsilon}}_{\text{obs},1}\}$ and $\mathbb{E}\{L^{\boldsymbol{\varepsilon}}_{\text{obs},2}\}$ of $D^{\boldsymbol{\varepsilon}}_{\text{obs}}$, $L^{\boldsymbol{\varepsilon}}_{\text{obs},1}$ and $L^{\boldsymbol{\varepsilon}}_{\text{obs},2}$ are known and given by $\delta^{\boldsymbol{\varepsilon}}_{\text{obs}}$, $\ell^{\boldsymbol{\varepsilon}}_{\text{obs},1}$ and $\ell^{\boldsymbol{\varepsilon}}_{\text{obs},2}$, respectively, that are $\mathbb{E}\{D^{\boldsymbol{\varepsilon}}_{\text{obs}}\} = \delta^{\boldsymbol{\varepsilon}}_{\text{obs}}$, $\mathbb{E}\{L^{\boldsymbol{\varepsilon}}_{\text{obs},1}\} = \ell^{\boldsymbol{\varepsilon}}_{\text{obs},1}$ and $\mathbb{E}\{L^{\boldsymbol{\varepsilon}}_{\text{obs},2}\} = \ell^{\boldsymbol{\varepsilon}}_{\text{obs},2}$, (iv) $[\boldsymbol{S}^{\text{eff}}_{\text{obs}}]$ is a second-order a.s. positive-definite symmetric real-valued random matrix whose mean value $\mathbb{E}\{[\boldsymbol{S}^{\text{eff}}_{\text{obs}}]\}$ is known and given by the positive-definite symmetric real matrix $[\underline{S}^{\text{eff}}_{\text{obs}}]$, that is $\mathbb{E}\{[\boldsymbol{S}^{\text{eff}}_{\text{obs}}]\} = [\underline{S}^{\text{eff}}_{\text{obs}}]$, and whose inverse matrix $[\boldsymbol{S}^{\text{eff}}_{\text{obs}}]^{-1}$ is a second-order random matrix. Then, the MaxEnt principle leads to statistically independent gamma (positive-valued) random variables $D^{\boldsymbol{\varepsilon}}_{\text{obs}}$, $L^{\boldsymbol{\varepsilon}}_{\text{obs},1}$ and $L^{\boldsymbol{\varepsilon}}_{\text{obs},2}$ whose mean values are known and given by $\delta^{\boldsymbol{\varepsilon}}_{\text{obs}}$, $\ell^{\boldsymbol{\varepsilon}}_{\text{obs},1}$ and $\ell^{\boldsymbol{\varepsilon}}_{\text{obs},2}$, respectively, and whose levels of statistical fluctuations are given by the unknown positive dispersion parameters s_0, s_1 and s_2, respectively, corresponding to the coefficients of variation of each of the random variables $D^{\boldsymbol{\varepsilon}}_{\text{obs}}$, $L^{\boldsymbol{\varepsilon}}_{\text{obs},1}$ and $L^{\boldsymbol{\varepsilon}}_{\text{obs},2}$. As a consequence, dispersion parameters s_0, s_1 and s_2 directly



(a) $p_{H_1}(h_1)$ (b) $p_{H_2}(h_2)$ (c) $p_{H_3}(h_3)$ (d) $p_{H_4}(h_4)$

Figure 14: Probability density functions (pdfs) p_{H_1}, p_{H_2}, p_{H_3} and p_{H_4} of random variables H_1, H_2, H_3 and H_4, respectively, with the uniform target pdfs (green), the estimated target pdfs (red), and the estimated output pdfs computed by using the best neural network trained with the initial database (black) and with the processed database (blue)

control the level of uncertainties of the 3 first random quantities of interest $Q_1^{\rm obs} = D_{\rm obs}^{\boldsymbol{\varepsilon}}$, $Q_2^{\rm obs} = L_{\rm obs,1}^{\boldsymbol{\varepsilon}}$ and $Q_3^{\rm obs} = L_{\rm obs,2}^{\boldsymbol{\varepsilon}}$, respectively. The MaxEnt principle also leads to a random matrix $[\boldsymbol{S}_{\rm obs}^{\rm eff}]$ in the ensemble SE_0^+ of non-Gaussian second-order a.s. positive-definite symmetric real random matrices [68, 69, 32] with a known mean value given by $[\underline{S}_{\rm obs}^{\rm eff}]$ and parameterized by an unknown positive dispersion parameter $s^{\rm eff}$ that is proportional to the coefficient of variation of $[\boldsymbol{S}_{\rm obs}^{\rm eff}]$ (see [1, 2, 32]), thus characterizing the level of statistical fluctuations of $[\boldsymbol{S}_{\rm obs}^{\rm eff}]$ around its mean value $[\underline{S}_{\rm obs}^{\rm eff}]$. As a consequence, dispersion parameter $s^{\rm eff}$ indirectly controls the level of uncertainties of the 6 last random quantities of interest $Q_4^{\rm obs} = \log([\boldsymbol{L}_{\rm obs}^{\rm eff}]_{11})$, $Q_5^{\rm obs} = [\boldsymbol{L}_{\rm obs}^{\rm eff}]_{12}$, $Q_6^{\rm obs} = [\boldsymbol{L}_{\rm obs}^{\rm eff}]_{13}$, $Q_7^{\rm obs} = \log([\boldsymbol{L}_{\rm obs}^{\rm eff}]_{22})$, $Q_8^{\rm obs} = [\boldsymbol{L}_{\rm obs}^{\rm eff}]_{23}$ and $Q_9^{\rm obs} = \log([\boldsymbol{L}_{\rm obs}^{\rm eff}]_{33})$. Finally, the prior probabilistic model of $\boldsymbol{Q}^{\rm obs}$ depends on the four-dimensional vector-valued hyperparameter $\boldsymbol{s} = (s_0, s_1, s_2, s^{\rm eff}) \in \,]0\,, +\infty[^4$ allowing the level of statistical fluctuations of random vector $\boldsymbol{Q}^{\rm obs}$ to be controlled. If experimental data are available on input random vector $\boldsymbol{Q}^{\rm obs}$, then an estimation of $\boldsymbol{s}$ can be carried out, *e.g.* by using the least-squares method [70, 32] or the maximum likelihood estimation method [71, 72, 57, 32]. If only one input vector $\boldsymbol{q}^{\rm obs}$ of observed quantities of interest is available, then $\boldsymbol{s}$ can be used to perform a sensitivity analysis of the network output value $\boldsymbol{h}^{\rm out}$ of hyperparameters. It is worth pointing out that the fundamental problem related to the identification of $\boldsymbol{s}$ is a challenging task that falls out of the scope of the present paper and is therefore left for future works. In the numerical examples presented in Sections 10 and 11, the robustness analysis of the network output with respect to $\boldsymbol{s}$ has been performed

by considering the same value s for each component of $\boldsymbol{s}$, that is $s_0 = s_1 = s_2 = s^{\text{eff}} = s$, and for different values of s arbitrarily chosen between 1% and 5% in order to provide a simple illustration of the proposed methodology. Such a probabilistic model of the random vector $\boldsymbol{Q}^{\text{obs}}$ of observed quantities of interest then allows the robustness of the output value $\boldsymbol{h}^{\text{out}}$ of hyperparameters to be analyzed with respect to the level of statistical fluctuations of $\boldsymbol{Q}^{\text{obs}}$ controlled by $\boldsymbol{s}$. The network output $\boldsymbol{h}^{\text{out}}$ is then modeled as a random variable $\boldsymbol{H}^{\text{out}} = (H_1^{\text{out}}, H_2^{\text{out}}, H_3^{\text{out}}, H_4^{\text{out}})$ defined in using (9) and such that

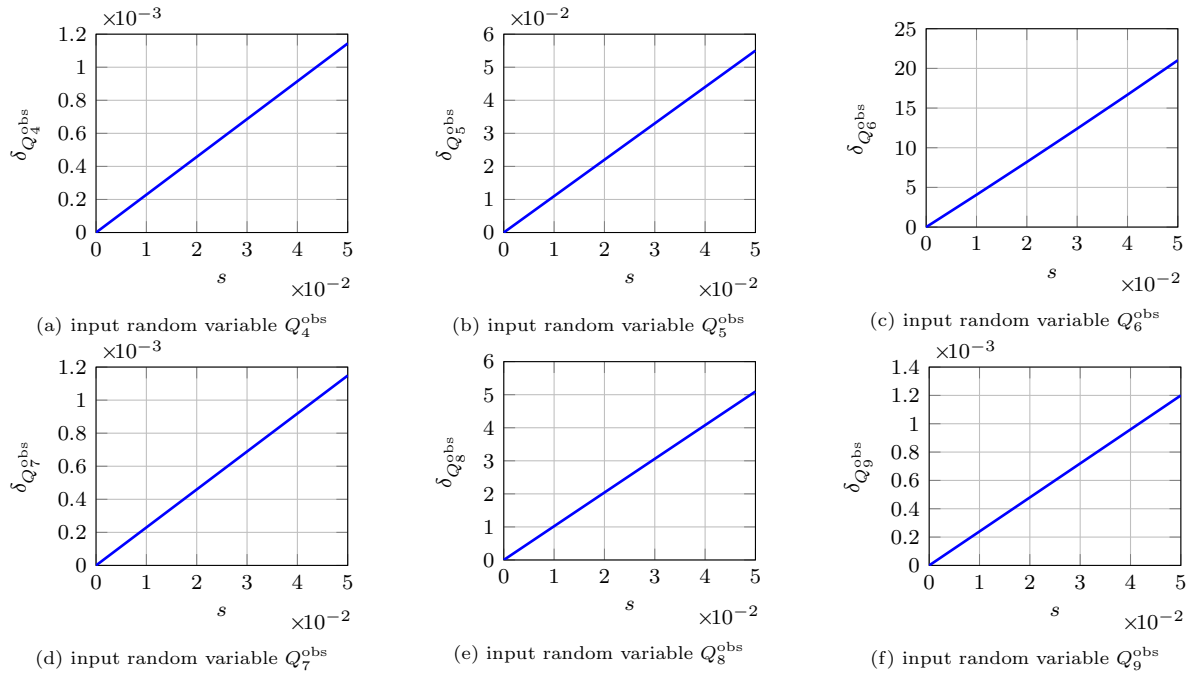
$$\boldsymbol{H}^{\text{out}} = \boldsymbol{\mathcal{N}}(\boldsymbol{Q}^{\text{obs}}). \tag{10}$$

Let $\boldsymbol{q}^{\text{obs},(1)}, \ldots, \boldsymbol{q}^{\text{obs},(N_s)}$ be N_s independent realizations of $\boldsymbol{Q}^{\text{obs}}$, then N_s independent realizations $\boldsymbol{h}^{\text{out},(1)}, \ldots, \boldsymbol{h}^{\text{out},(N_s)}$ of $\boldsymbol{H}^{\text{out}}$ are constructed in using (10) and we then have $\boldsymbol{h}^{\text{out},(i)} = \boldsymbol{\mathcal{N}}(\boldsymbol{q}^{\text{obs},(i)})$ for $i = 1, \ldots, N_s$. For identification purposes in the presence of experimental errors, the solution $\boldsymbol{h}^*$ of the underlying statistical inverse problem can be defined as the output mean value $\underline{\boldsymbol{h}}^{\text{out}} = \mathbb{E}\{\boldsymbol{H}^{\text{out}}\}$ estimated in using the N_s independent realizations $\boldsymbol{h}^{\text{out},(1)}, \ldots, \boldsymbol{h}^{\text{out},(N_s)}$ of $\boldsymbol{H}^{\text{out}}$ with the mathematical statistics [71]. In addition, the network outputs can be used for constructing the marginal probability density functions $p_{H_1^{\text{out}}}$, $p_{H_2^{\text{out}}}$, $p_{H_3^{\text{out}}}$ and $p_{H_4^{\text{out}}}$ of the components of output random vector $\boldsymbol{H}^{\text{out}} = (H_1^{\text{out}}, H_2^{\text{out}}, H_3^{\text{out}}, H_4^{\text{out}})$ by using the univariate Gaussian kernel density estimation method [28] in order to quantify the robustness of the output vectors of hyperparameters generated by the trained neural network with respect to some experimental errors on the input vector of observed quantities of interest.

10. Numerical example on synthetic data

Based on the numerical results obtained in Section 8.2, we consider the best three-layer neural network trained with the processed database for identification purposes. Hereinafter, the neural network-based identification method is first applied to synthetic data coming from numerical simulations and then carried out on real experimental data coming from experimental measurements on a bovine cortical bone specimen in Section 11.

We first consider a given input vector $\boldsymbol{q}^{\text{obs}}$ of quantities of interest contained in the test dataset for validating the proposed neural network-based identification method. The network output vector $\boldsymbol{h}^{\text{out}}$ is directly computed by using the trained neural network with $\boldsymbol{q}^{\text{obs}}$ as input vector and compared to the corresponding target vector $\boldsymbol{h}^{\text{target}}$. For analyzing the robustness of the output vector $\boldsymbol{h}^{\text{out}}$ of hyperparameters with respect to the uncertainties on the input vector $\boldsymbol{q}^{\text{obs}}$ of observed quantities of interest, $N_s = 10^6$ independent realizations $\boldsymbol{q}^{\text{obs},(1)}, \ldots, \boldsymbol{q}^{\text{obs},(N_s)}$ of input random vector $\boldsymbol{Q}^{\text{obs}}$ are generated according to its probabilistic model presented in Section 9 and parameterized by the vector-valued parameter $\boldsymbol{s} = (s_0, s_1, s_2, s^{\text{eff}})$ controlling the level of statistical fluctuations of $\boldsymbol{Q}^{\text{obs}}$ around its mean value $\boldsymbol{q}^{\text{obs}}$. The best trained neural network is then used for simulating the corresponding N_s independent realizations $\boldsymbol{h}^{\text{out},(1)}, \ldots, \boldsymbol{h}^{\text{out},(N_s)}$ of output random vector $\boldsymbol{H}^{\text{out}}$, from which the mean value $\underline{\boldsymbol{h}}^{\text{out}} = \mathbb{E}\{\boldsymbol{H}^{\text{out}}\}$ and the confidence interval I^{out} with a probability level 95% of $\boldsymbol{H}^{\text{out}}$ are estimated by using the mathematical statistics. In order to quantify the robustness of the best trained neural network with respect to the uncertainties on input vector $\boldsymbol{q}^{\text{obs}}$, we consider the same input uncertainty level s for each of the components of $\boldsymbol{s}$, that is $s_0 = s_1 = s_2 = s^{\text{eff}} = s$, and we perform a robustness analysis of the network output mean vector $\underline{\boldsymbol{h}}^{\text{out}}$ with respect to the input uncertainty level s by considering increasing values for $s \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$. Recall that $\underline{\boldsymbol{h}}^{\text{out}}$ *a priori* coincides with $\boldsymbol{h}^{\text{out}}$ only when $s = 0$ (*i.e.* in the absence of uncertainties on input vector $\boldsymbol{q}^{\text{obs}}$). According to Section 9, the value of s corresponds to the coefficient of variation of each of the 3 first components $Q_1^{\text{obs}} = D_{\text{obs}}^{\boldsymbol{\varepsilon}}$, $Q_2^{\text{obs}} = L_{\text{obs},1}^{\boldsymbol{\varepsilon}}$ and $Q_3^{\text{obs}} = L_{\text{obs},2}^{\boldsymbol{\varepsilon}}$ of random vector $\boldsymbol{Q}^{\text{obs}}$ and therefore allows the level of statistical fluctuations of these input random variables (around their respective mean values) to be directly controlled. Also, the value of s is proportional to the coefficient of variation of $[\boldsymbol{S}_{\text{obs}}^{\text{eff}}]$ (see [1, 2, 32]) and therefore allows the level of statistical fluctuations of the 6 last components Q_4^{obs}, Q_5^{obs}, Q_6^{obs}, Q_7^{obs}, Q_8^{obs} and Q_9^{obs} of random vector $\boldsymbol{Q}^{\text{obs}}$ (around their respective mean values) to be indirectly controlled. Figure 15 shows the evolutions of the coefficient of variation $\delta_{Q_i^{\text{obs}}}$ of each component Q_i^{obs} of random vector $\boldsymbol{Q}^{\text{obs}}$, for $i = 4, \ldots, 9$, with respect to s in order to quantify the impact of dispersion parameter s on the network input random variables $Q_4^{\text{obs}}, \ldots, Q_9^{\text{obs}}$. For each input random variable Q_i^{obs}, the coefficient of variation $\delta_{Q_i^{\text{obs}}}$ increases linearly with input uncertainty level s.



(a) input random variable Q_4^{obs} (b) input random variable Q_5^{obs} (c) input random variable Q_6^{obs}

(d) input random variable Q_7^{obs} (e) input random variable Q_8^{obs} (f) input random variable Q_9^{obs}

Figure 15: Synthetic data: evolutions of the coefficient of variation $\delta_{Q_i^{\text{obs}}}$ of each network input random variable Q_i^{obs} estimated from the $N_s = 10^6$ independent realizations $q_i^{\text{obs},(1)}, \ldots, q_i^{\text{obs},(N_s)}$, for $i = 4, \ldots, 9$, with respect to input uncertainty level s

The identification results obtained with the trained neural network for the different input uncertainty levels are summarized in Table 3. In addition, Figure 16 displays the graphs of the mean output values $\underline{h}_1^{\text{out}}$, $\underline{h}_2^{\text{out}}$, $\underline{h}_3^{\text{out}}$ and $\underline{h}_4^{\text{out}}$ and the corresponding confidence regions (with a probability level 95%) of H_1^{out}, H_2^{out}, H_3^{out} and H_4^{out}, respectively, with respect to the input uncertainty level s. First, in the absence of uncertainty on input vector $\boldsymbol{q}^{\text{obs}}$ (*i.e.* for an input uncertainty level $s = 0$), the values of output vector $\boldsymbol{h}^{\text{out}}$ computed by using the trained neural network are very close to the corresponding values of target vector $\boldsymbol{h}^{\text{target}}$ with relative errors less than 1% for each of the random hyperparameters H_1, H_2, H_3 and H_4. Secondly, in the presence of uncertainties on input vector $\boldsymbol{q}^{\text{obs}}$, the values of output mean vector $\underline{\boldsymbol{h}}^{\text{out}}$ remain close to the corresponding values of target vector $\boldsymbol{h}^{\text{target}}$ with maximum relative errors less than 0.3%, 2%, 3% and 3% for the mean output values $\underline{h}_1^{\text{out}}$, $\underline{h}_2^{\text{out}}$, $\underline{h}_3^{\text{out}}$ and $\underline{h}_4^{\text{out}}$, respectively, for the highest input uncertainty level $s = 0.05 = 5\%$ considered here. Thus, even though the output 95% confidence intervals become wider as the input uncertainty level s increases, the values of output mean vector $\underline{\boldsymbol{h}}^{\text{out}}$ present small variations with respect to the input uncertainty level s.

Figure 17 shows the marginal probability density functions $p_{H_1^{\text{out}}}$, $p_{H_2^{\text{out}}}$, $p_{H_3^{\text{out}}}$ and $p_{H_4^{\text{out}}}$ of random variables H_1^{out}, H_2^{out}, H_3^{out} and H_4^{out}, respectively, estimated by using the univariate Gaussian kernel density estimation method with the $N_s = 10^6$ independent realizations $\boldsymbol{h}^{\text{out},(1)}, \ldots, \boldsymbol{h}^{\text{out},(N_s)}$ of $\boldsymbol{H}^{\text{out}} = (H_1^{\text{out}}, H_2^{\text{out}}, H_3^{\text{out}}, H_4^{\text{out}})$ for a given input uncertainty level $s = 0.01 = 1\%$. For such a small dispersion on the input values, the output values generated by the trained neural network present small fluctuations and remain concentrated around the output mean value, and the associated target value lies within the output 95% confidence interval, for each of the random variables H_1^{out}, H_2^{out}, H_3^{out} and H_4^{out}. Figure 18 represents the marginal probability density functions $p_{H_1^{\text{out}}}$, $p_{H_2^{\text{out}}}$, $p_{H_3^{\text{out}}}$ and $p_{H_4^{\text{out}}}$ for the different values of input uncertainty level s ranging from $0.01 = 1\%$ to $0.05 = 5\%$. For each component H_j^{out} of output random vector $\boldsymbol{H}^{\text{out}}$, the higher the uncertainty level s, the more flattened the probability density function $p_{H_j^{\text{out}}}$ is, but the mean value $\underline{h}_j^{\text{out}}$ of output random variable H_j^{out} still remains a good approximation of the target value h_j^{target} even with an input level of uncertainties $s = 0.05 = 5\%$ on observed input vector $\boldsymbol{q}^{\text{obs}}$. The proposed neural network-based identification method then remains accurate even in the

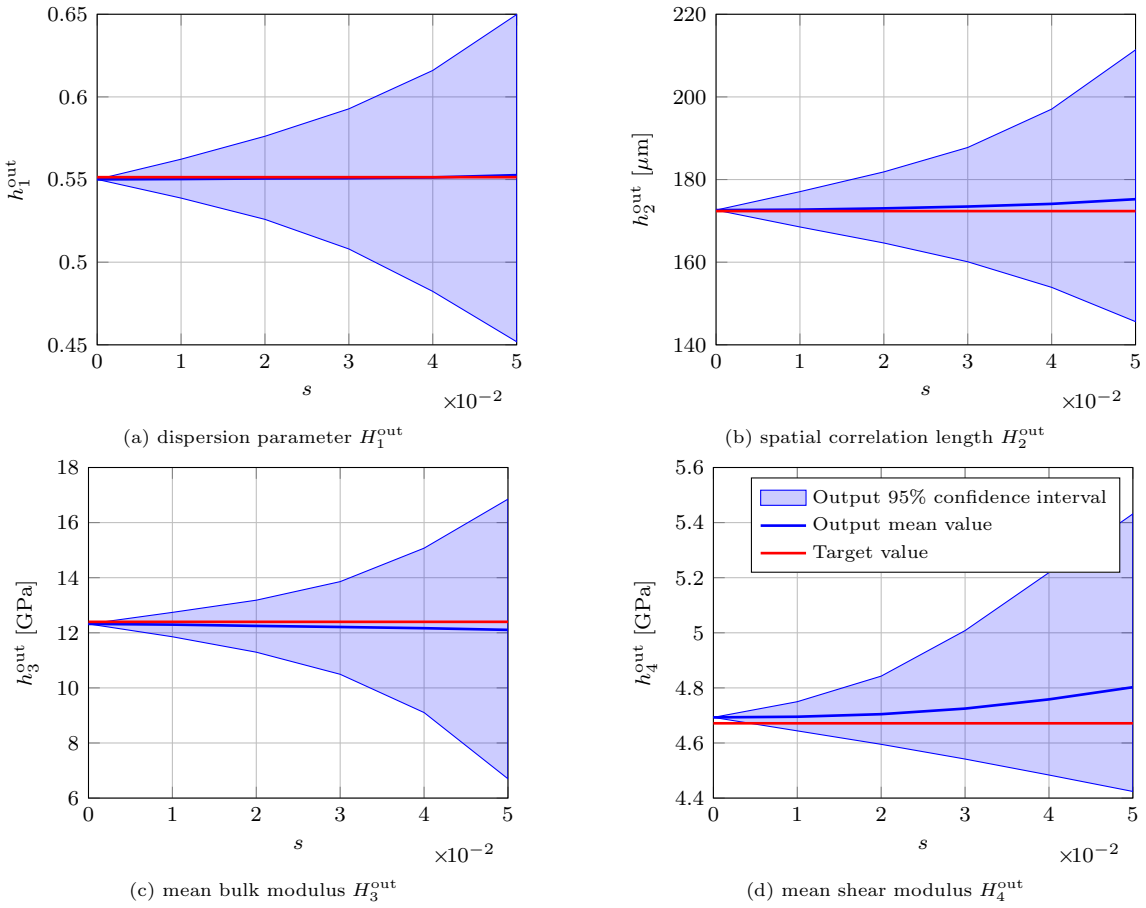
Table 3: Synthetic data: comparison of output vector $\boldsymbol{h}^{\text{out}}$ and output mean vector $\underline{\boldsymbol{h}}^{\text{out}}$ with associated target vector $\boldsymbol{h}^{\text{target}}$, and output 95% confidence interval I^{out} obtained for a given input vector $\boldsymbol{q}^{\text{obs}}$ contained in the test dataset and for different values of input uncertainty level s

		h_1	h_2 [μm]	h_3 [GPa]	h_4 [GPa]
Target vector $\boldsymbol{h}^{\text{target}}$		0.5514	172.36	12.398	4.672
Output vector $\boldsymbol{h}^{\text{out}}$	$s = 0\%$	0.5501	172.61	12.322	4.693
Output mean vector $\underline{\boldsymbol{h}}^{\text{out}}$	$s = 1\%$	0.5503	172.71	12.297	4.695
	$s = 2\%$	0.5507	173.03	12.250	4.705
	$s = 3\%$	0.5508	173.47	12.211	4.725
	$s = 4\%$	0.5513	174.11	12.167	4.758
	$s = 5\%$	0.5527	175.25	12.106	4.802
Relative error [%]	$s = 0\%$	0.24	0.14	0.61	0.46
	$s = 1\%$	0.20	0.20	0.81	0.51
	$s = 2\%$	0.14	0.39	1.19	0.71
	$s = 3\%$	0.11	0.64	1.50	1.14
	$s = 4\%$	0.02	1.01	1.86	1.86
	$s = 5\%$	0.22	1.67	2.35	2.80
Output 95% confidence interval I^{out}	$s = 1\%$	[0.5388 , 0.5623]	[168.53 , 177.07]	[11.856 , 12.741]	[4.644 , 4.750]
	$s = 2\%$	[0.5259 , 0.5762]	[164.65 , 181.86]	[11.296 , 13.184]	[4.595 , 4.843]
	$s = 3\%$	[0.5080 , 0.5928]	[160.10 , 187.76]	[10.494 , 13.858]	[4.542 , 5.008]
	$s = 4\%$	[0.4823 , 0.6160]	[153.91 , 197.04]	[9.105 , 15.072]	[4.484 , 5.218]
	$s = 5\%$	[0.4519 , 0.6498]	[145.62 , 211.39]	[6.700 , 16.852]	[4.424 , 5.432]

presence of uncertainties on the given input vector $\boldsymbol{q}^{\text{obs}}$ of quantities of interest. It can therefore be applied to experimentally measured quantities of interest.

11. Numerical example on real experimental data for a biological material

We now consider a given input vector $\boldsymbol{q}^{\text{obs}}$ of observed quantities of interest coming from experimental measurements of 2D displacement fields obtained from a single static vertical uniaxial compression test performed on a unique cubic specimen (with dimensions $1 \times 1 \times 1$ cm^3) made of a biological tissue (beef femur cortical bone) and monitored by 2D digital image correlation (DIC) on one observed side of the cubic specimen corresponding to a 2D square domain Ω^{macro} with macroscopic dimensions 1×1 cm^2. Such experimental kinematic field measurements have been carried out in [48] and already used in [3, 4] for identifying the apparent elastic properties of bovine cortical bone at mesoscale. The experimental test configuration corresponds to the numerical one described in Figure 4. The interested reader can refer to [48] for technical details concerning the experimental setup of the mechanical test (specimen preparation, test bench, test procedure, optical measuring instrument, optical image acquisition system and DIC method) for obtaining the 2D displacement field measurements. The experimental quantities of interest $\boldsymbol{q}^{\text{obs}} = (q_1^{\text{obs}}, \dots, q_9^{\text{obs}})$ have been derived from the experimental fine-scale displacement field computed on a 2D square subdomain $\Omega^{\text{meso}} \subset \Omega^{\text{macro}}$ with mesoscopic dimensions 1×1 cm^2 (located near the center of the observed face of the cubic sample to limit edge effects) and discretized with a fine regular grid of 100×100 quadrangular elements with uniform element size $h^{\text{meso}} = 10$ µm $= 10^{-5}$ m in each spatial direction. The experimental linearized strain field has been directly computed from the experimentally measured displacement field by using classical interpolation techniques and then used to compute the three first experimental quantities of interest q_1^{obs}, q_2^{obs} and q_3^{obs}, where q_1^{obs} corresponds to the spatial dispersion coefficient quantifying the level of spatial fluctuations of the linearized experimental strain field around its spatial average over Ω^{meso}, while q_2^{obs} and q_3^{obs} correspond to the two characteristic lengths along the two spatial directions x_1 and x_2 characterizing the spatial fluctuations of the linearized experimental strain field around its spatial average over Ω^{meso} and numerically computed using a usual signal processing method. The effective compliance matrix



(a) dispersion parameter H_1^{out}

(b) spatial correlation length H_2^{out}

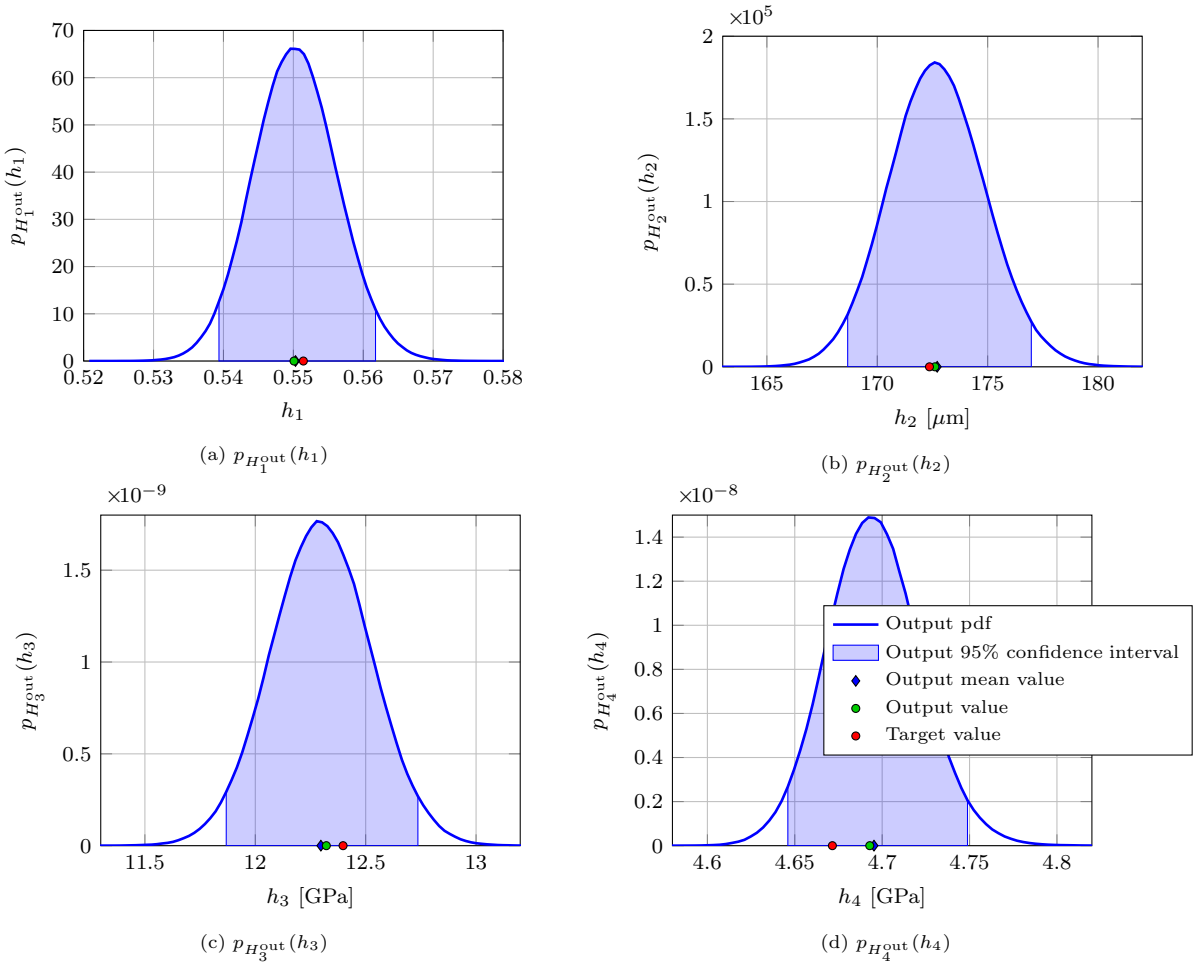
(c) mean bulk modulus H_3^{out}

(d) mean shear modulus H_4^{out}

Figure 16: Synthetic data: evolutions of the output mean values $\underline{\boldsymbol{h}}^{\text{out}}$ (blue curve) and the 95% confidence intervals I^{out} (blue areas) of random variables H_1^{out}, H_2^{out}, H_3^{out} and H_4^{out}, respectively, with respect to input uncertainty level s, obtained for a given input vector $\boldsymbol{q}^{\text{obs}}$, with the corresponding target values $\boldsymbol{h}^{\text{target}}$ (red lines)

$[S_{\text{obs}}^{\text{eff}}] \in \mathbb{M}_3^+(\mathbb{R})$ has experimentally been identified in previous works [3, 4] by solving a classical inverse problem at coarse scale (macroscale) using experimental coarse-scale displacement field measurements at macroscale. More precisely, since the observed face of the cubic sample corresponds to a plane of isotropy of the material, $[S_{\text{obs}}^{\text{eff}}]$ is completely characterized and parameterized by the bulk modulus κ and the shear modulus μ of the isotropic elastic material at macroscale. The optimal values of κ and μ have been identified by minimizing the spatial average over macroscopic domain Ω^{macro} of the distance (defined with respect to the Frobenius norm) between the strain field (parameterized by (κ, μ)) computed numerically by solving the deterministic linear elasticity boundary value problem (that models the experimental test configuration) at macroscale and the strain field measured experimentally at macroscale. The upper Cholesky factor $[L_{\text{obs}}^{\text{eff}}]$ of $[S_{\text{obs}}^{\text{eff}}]$ has then been computed by performing the Cholesky factorization of $[S_{\text{obs}}^{\text{eff}}] = [L_{\text{obs}}^{\text{eff}}]^T [L_{\text{obs}}^{\text{eff}}]$ in order to derive the last vector-valued experimental quantity of interest $(q_4^{\text{obs}}, \ldots, q_9^{\text{obs}})$, with $q_4^{\text{obs}} = \log([L_{\text{obs}}^{\text{eff}}]_{11})$, $q_5^{\text{obs}} = [L_{\text{obs}}^{\text{eff}}]_{12}$, $q_6^{\text{obs}} = [L_{\text{obs}}^{\text{eff}}]_{13}$, $q_7^{\text{obs}} = \log([L_{\text{obs}}^{\text{eff}}]_{22})$, $q_8^{\text{obs}} = [L_{\text{obs}}^{\text{eff}}]_{23}$ and $q_9^{\text{obs}} = \log([L_{\text{obs}}^{\text{eff}}]_{33})$, as presented in Section 2.

As for the previous validation example on synthetic data, the trained neural network is first used to compute the output vector $\boldsymbol{h}^{\text{out}}$ for the experimentally observed input vector $\boldsymbol{q}^{\text{obs}}$ without introducing uncertainties. Then, in order to quantify the robustness of the network output vector $\boldsymbol{h}^{\text{out}}$ with respect to the uncertainties on the input vector $\boldsymbol{q}^{\text{obs}}$, we consider the input random vector $\boldsymbol{Q}^{\text{obs}}$ whose probabilistic model has been introduced in Section 9 and which is parameterized by the four-dimensional vector-valued



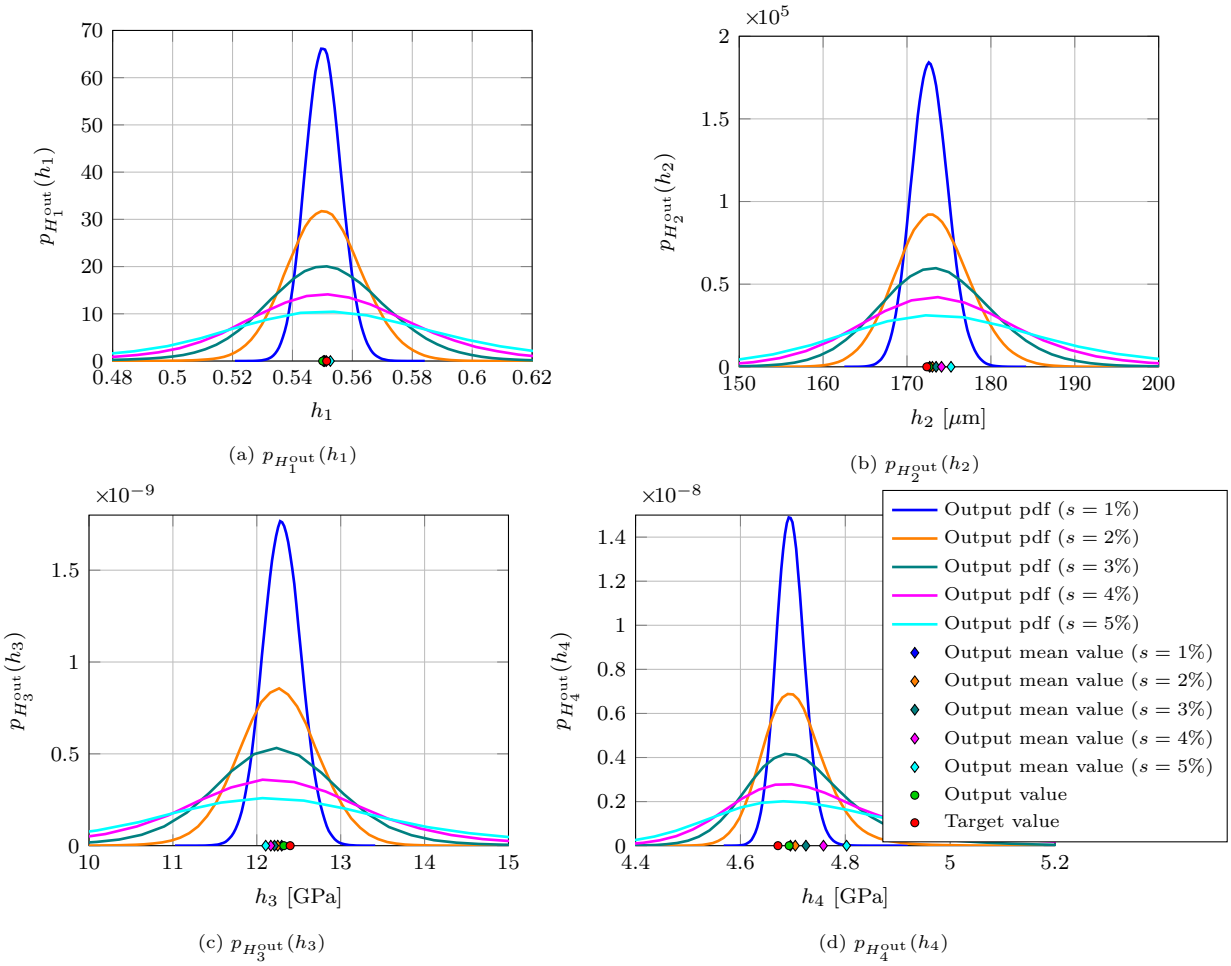
(a) $p_{H_1^{\text{out}}}(h_1)$ (b) $p_{H_2^{\text{out}}}(h_2)$

(c) $p_{H_3^{\text{out}}}(h_3)$ (d) $p_{H_4^{\text{out}}}(h_4)$

Figure 17: Synthetic data: probability density functions $p_{H_1^{\text{out}}}$, $p_{H_2^{\text{out}}}$, $p_{H_3^{\text{out}}}$ and $p_{H_4^{\text{out}}}$ of random variables H_1^{out}, H_2^{out}, H_3^{out} and H_4^{out}, respectively, obtained for a given input vector $\boldsymbol{q}^{\text{obs}}$ and for a given input uncertainty level $s = 0.01 = 1\%$, with the output 95% confidence intervals I^{out} (blue areas), the output mean values $\underline{\boldsymbol{h}}^{\text{out}}$ (blue diamonds), the output values $\boldsymbol{h}^{\text{out}}$ (green circles) and the corresponding target values $\boldsymbol{h}^{\text{target}}$ (red circles)

parameter $\boldsymbol{s} = (s_0, s_1, s_2, s^{\text{eff}})$ with $s_0 = s_1 = s_2 = s^{\text{eff}} = s$, in which s is the input uncertainty level allowing the level of statistical fluctuations of $\boldsymbol{Q}^{\text{obs}}$ around its mean value $\boldsymbol{q}^{\text{obs}}$ to be controlled. In practice, the value of s is related to the knowledge of the experimental errors and should be driven by the expertise of the experimenter. For the considered application on real bovine cortical bone data, a reasonable value for s is of the order of few percents. In the following, we consider five different values for input uncertainty level $s \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$ and for each of them, $N_s = 10^6$ independent realizations $\boldsymbol{q}^{\text{obs},(1)}, \ldots, \boldsymbol{q}^{\text{obs},(N_s)}$ of input random vector $\boldsymbol{Q}^{\text{obs}}$ are generated, then presented and applied to the trained neural network in order to compute the N_s corresponding independent realizations $\boldsymbol{h}^{\text{out},(1)}, \ldots, \boldsymbol{h}^{\text{out},(N_s)}$ of output random vector $\boldsymbol{H}^{\text{out}}$. The values of output mean vector $\boldsymbol{h}^{\text{out}}$ and the bounds of the confidence intervals of each of the components of $\boldsymbol{H}^{\text{out}}$ are then computed by using classical empirical estimates.

Table 4 reports the values of output vector $\boldsymbol{h}^{\text{out}}$ (corresponding to an input uncertainty level $s = 0$) and the ones of output mean vector $\underline{\boldsymbol{h}}^{\text{out}}$ as well as the bounds of the output 95% confidence intervals of $\boldsymbol{H}^{\text{out}}$ for the different values of input uncertainty level $s \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$. As a complement, Figure 19 represents the graphs of the mean output values $\underline{h}_1^{\text{out}}$, $\underline{h}_2^{\text{out}}$, $\underline{h}_3^{\text{out}}$ and $\underline{h}_4^{\text{out}}$ and the corresponding confidence regions (with a probability level 95%) of H_1^{out}, H_2^{out}, H_3^{out} and H_4^{out}, respectively, with respect to the input uncertainty level s. We observe that the mean value $\underline{h}_1^{\text{out}}$ of H_1^{out} is relatively insensitive to the



(a) $p_{H_1^{\text{out}}}(h_1)$

(b) $p_{H_2^{\text{out}}}(h_2)$

(c) $p_{H_3^{\text{out}}}(h_3)$

(d) $p_{H_4^{\text{out}}}(h_4)$

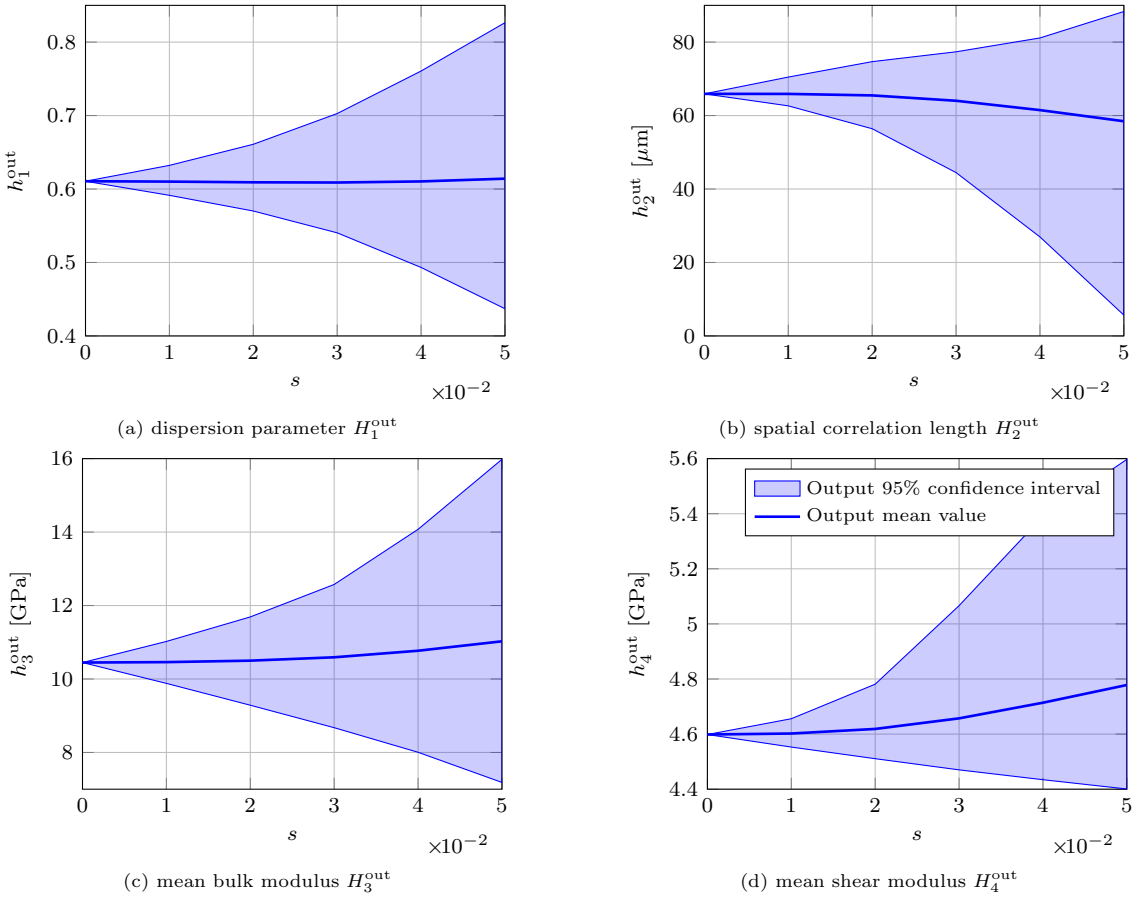
Figure 18: Synthetic data: probability density functions $p_{H_1^{\text{out}}}$, $p_{H_2^{\text{out}}}$, $p_{H_3^{\text{out}}}$ and $p_{H_4^{\text{out}}}$ of random variables H_1^{out}, H_2^{out}, H_3^{out} and H_4^{out}, respectively, obtained for a given input vector $\boldsymbol{q}^{\text{obs}}$ and for different values of input uncertainty level $s \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$, with the output mean values $\underline{\boldsymbol{h}}^{\text{out}}$ (colored diamonds), the output values $\boldsymbol{h}^{\text{out}}$ (green circles) and the corresponding target values $\boldsymbol{h}^{\text{target}}$ (red circles)

input uncertainty level s, while the mean values $\underline{h}_2^{\text{out}}$, $\underline{h}_3^{\text{out}}$ and $\underline{h}_4^{\text{out}}$ of H_2^{out}, H_3^{out} and H_4^{out} slightly vary and deviate from the respective output values h_2^{out}, h_3^{out} and h_4^{out} (obtained by using the trained neural network with an input uncertainty level $s = 0$) as the input uncertainty level s increases. It should be noted that the values of the output vector $\boldsymbol{h}^{\text{out}} = (0.6106, 65.906, 10.448, 4.598)$ in $[-]\times[\mu\text{m}]\times[\text{GPa}]\times[\text{GPa}]$ obtained for the input vector $\boldsymbol{q}^{\text{obs}}$ (without considering any input uncertainties) are close to the values of the optimal vector $\boldsymbol{h}^{\text{opt}} = (0.533, 61.111, 10.500, 4.667)$ in $([-], [\mu\text{m}], [\text{GPa}], [\text{GPa}])$ obtained in the previous work [4] by solving a computationally expensive multi-objective optimization problem using a fixed-point iterative algorithm with the same experimental measurements as those used in the present work. The identified values obtained with the previous method in [4] result from a compromise between computational efficiency and numerical accuracy and are therefore less accurate than the ones obtained with the ANN-based identification method proposed in this work. The network output values are then in agreement with the identified values already published in the literature for this type of biological tissue (bovine cortical bone).

The marginal probability density functions $p_{H_1^{\text{out}}}$, $p_{H_2^{\text{out}}}$, $p_{H_3^{\text{out}}}$ and $p_{H_4^{\text{out}}}$ of random variables H_1^{out}, H_2^{out}, H_3^{out} and H_4^{out}, respectively, estimated by using the kernel density estimation method with the $N_s = 10^6$ independent realizations $\boldsymbol{h}^{\text{out},(1)}, \ldots, \boldsymbol{h}^{\text{out},(N_s)}$ of $\boldsymbol{H}^{\text{out}} = (H_1^{\text{out}}, H_2^{\text{out}}, H_3^{\text{out}}, H_4^{\text{out}})$ are represented in Figure 20 for a given input uncertainty level $s = 0.01 = 1\%$ and in Figure 21 for an input uncertainty

Table 4: Real experimental data: output vector $\boldsymbol{h}^{\text{out}}$, output mean vector $\underline{\boldsymbol{h}}^{\text{out}}$ and output 95% confidence interval I^{out} obtained for a given input vector $\boldsymbol{q}^{\text{obs}}$ and for different values of input uncertainty level s

		h_1	h_2 [μm]	h_3 [GPa]	h_4 [GPa]
Output vector $\boldsymbol{h}^{\text{out}}$	$s = 0\%$	0.6106	65.906	10.448	4.598
Output mean vector $\underline{\boldsymbol{h}}^{\text{out}}$	$s = 1\%$	0.6101	65.891	10.460	4.602
	$s = 2\%$	0.6091	65.485	10.499	4.619
	$s = 3\%$	0.6089	64.026	10.591	4.657
	$s = 4\%$	0.6103	61.484	10.770	4.714
	$s = 5\%$	0.6141	58.456	11.026	4.778
Output 95% confidence interval I^{out}	$s = 1\%$	[0.5914 , 0.6322]	[62.640 , 70.474]	[9.881 , 11.022]	[4.553 , 4.656]
	$s = 2\%$	[0.5701 , 0.6609]	[56.412 , 74.675]	[9.286 , 11.690]	[4.511 , 4.781]
	$s = 3\%$	[0.5403 , 0.7027]	[44.472 , 77.355]	[8.674 , 12.574]	[4.470 , 5.066]
	$s = 4\%$	[0.4932 , 0.7606]	[26.954 , 81.128]	[8.005 , 14.075]	[4.434 , 5.391]
	$s = 5\%$	[0.4370 , 0.8264]	[5.665 , 88.325]	[7.184 , 15.980]	[4.401 , 5.597]



(a) dispersion parameter H_1^{out}

(b) spatial correlation length H_2^{out}

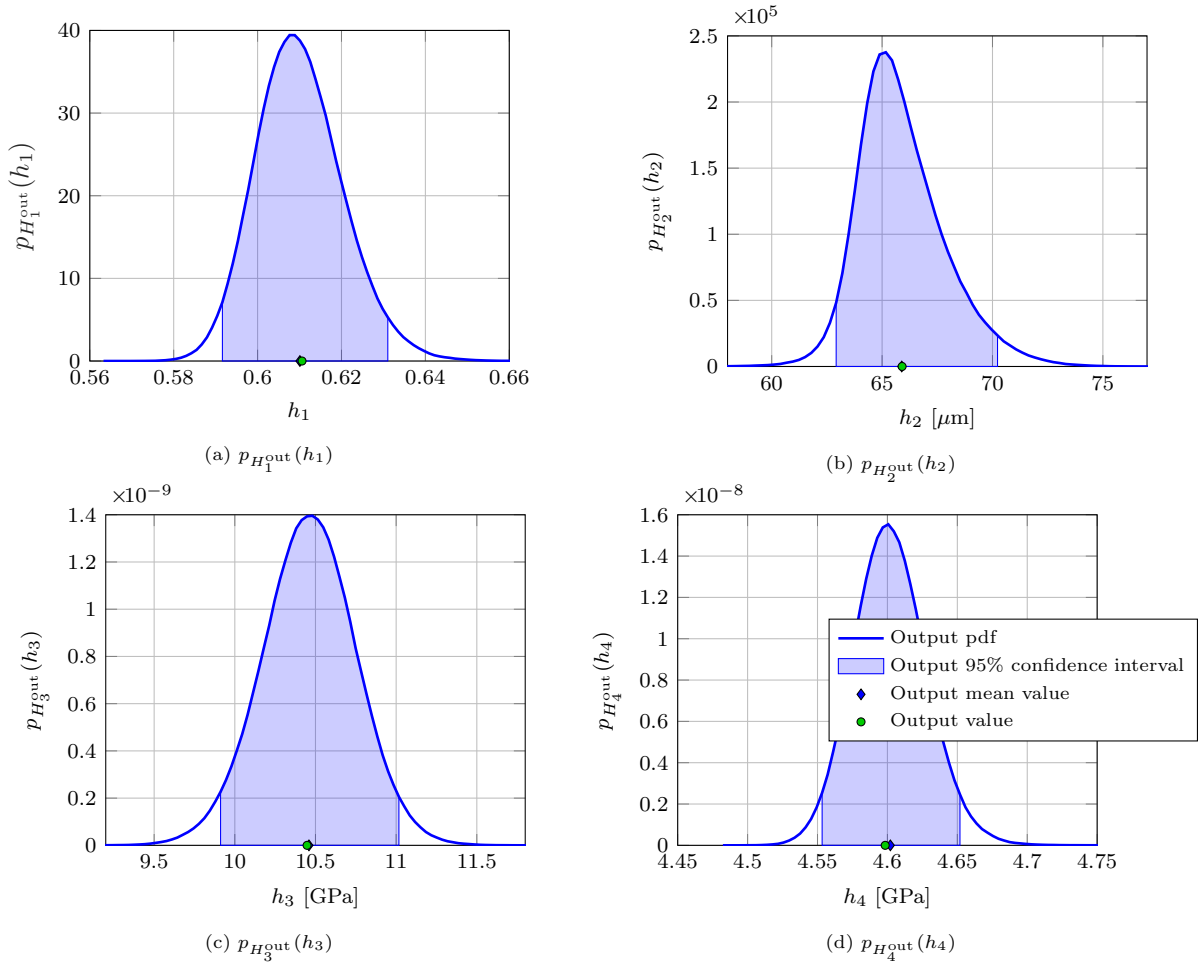
(c) mean bulk modulus H_3^{out}

(d) mean shear modulus H_4^{out}

Figure 19: Real experimental data: evolutions of the output mean values $\underline{\boldsymbol{h}}^{\text{out}}$ (blue curve) and the 95% confidence intervals I^{out} (blue areas) of random variables H_1^{out}, H_2^{out}, H_3^{out} and H_4^{out}, respectively, with respect to input uncertainty level s, for a given input vector $\boldsymbol{q}^{\text{obs}}$

level s varying from $0.01 = 1\%$ to $0.05 = 5\%$. We observe similar trends as for the previous validation example. Despite the large scattering of the network outputs for the highest input uncertainties level, the output mean values $\underline{h}_1^{\text{out}}$, $\underline{h}_2^{\text{out}}$, $\underline{h}_3^{\text{out}}$ and $\underline{h}_4^{\text{out}}$ are still relatively close to the corresponding output values

h_1^{out}, h_2^{out}, h_3^{out} and h_4^{out} (obtained without considering input uncertainties), thus showing the capability of the neural network-based identification method to efficiently computing robust output predictions with respect to the input uncertainties. Finally, such a real-world application has demonstrated the potential of the proposed identification method for solving the challenging statistical inverse problem related to the statistical identification of a stochastic model of the random compliance field (in high stochastic dimension) at mesoscale for a heterogeneous anisotropic elastic microstructure, by making use of a trained artificial neural network.



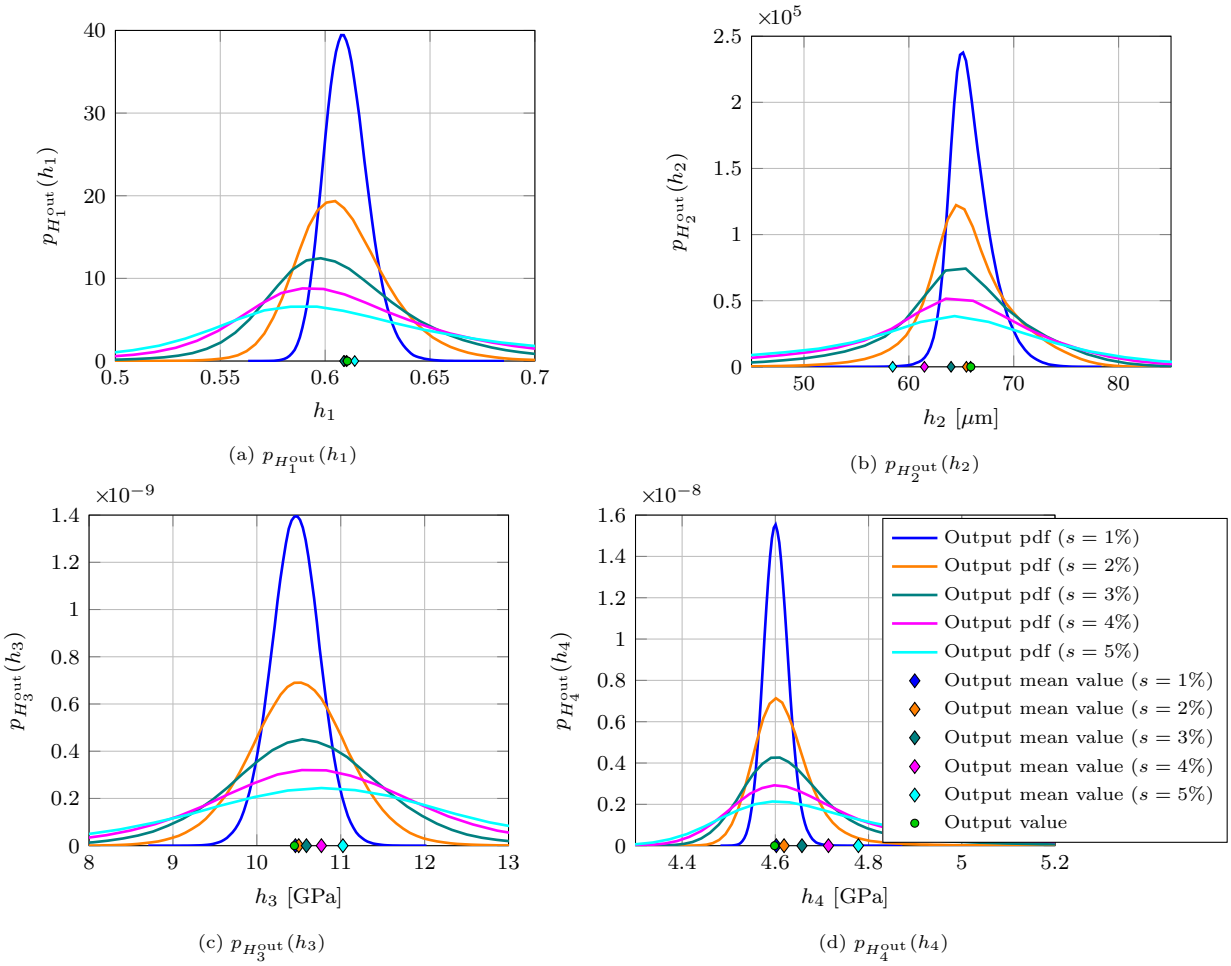
(a) $p_{H_1^{\text{out}}}(h_1)$ (b) $p_{H_2^{\text{out}}}(h_2)$

(c) $p_{H_3^{\text{out}}}(h_3)$ (d) $p_{H_4^{\text{out}}}(h_4)$

Figure 20: Real experimental data: probability density functions $p_{H_1^{\text{out}}}$, $p_{H_2^{\text{out}}}$, $p_{H_3^{\text{out}}}$ and $p_{H_4^{\text{out}}}$ of random variables H_1^{out}, H_2^{out}, H_3^{out} and H_4^{out}, respectively, obtained for a given input vector $\boldsymbol{q}^{\text{obs}}$ and for a given input uncertainty level $s = 0.01 = 1\%$, with the output 95% confidence intervals I^{out} (blue areas), the output mean values $\underline{\boldsymbol{h}}^{\text{out}}$ (blue diamonds) and the output values $\boldsymbol{h}^{\text{out}}$ (green circles)

12. Conclusions

In this paper, a neural network-based identification method has been presented for solving the statistical inverse problem related to the statistical identification of the hyperparameters of a prior stochastic model of the random compliance elasticity field characterizing the apparent elastic properties of heterogeneous materials with complex random microstructure. Such a challenging statistical inverse problem has been formulated as a function approximation problem and solved by using an artificial neural network trained from



(a) $p_{H_1^{\text{out}}}(h_1)$ (b) $p_{H_2^{\text{out}}}(h_2)$

(c) $p_{H_3^{\text{out}}}(h_3)$ (d) $p_{H_4^{\text{out}}}(h_4)$

Figure 21: Real experimental data: probability density functions $p_{H_1^{\text{out}}}$, $p_{H_2^{\text{out}}}$, $p_{H_3^{\text{out}}}$ and $p_{H_4^{\text{out}}}$ of random variables H_1^{out}, H_2^{out}, H_3^{out} and H_4^{out}, respectively, obtained for a given input vector $\boldsymbol{q}^{\text{obs}}$ and for different values of input uncertainty level $s \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$, with the output mean values $\underline{\boldsymbol{h}}^{\text{out}}$ (colored diamonds) and the output values $\boldsymbol{h}^{\text{out}}$ (green circles)

a large numerical database. A first (initial) database has been generated using forward numerical simulations of mechanical computational models introduced within the framework of linear elasticity theory under 2D plane stress assumption. A second (processed) database has been derived by conditioning the input data contained in the initial database with respect to the target data. A sensitivity analysis of the target data with respect to the input data contained in each of the two databases has been performed. Two- and three-layer feedforward neural networks have been trained with each of the initial and processed databases and optimized by considering different network configurations in order to construct fine-tuned trained models. Numerical results show that the neural networks trained with the processed database exhibit much better performances in terms of mean squared error, linear regression analysis and probability distribution between network outputs and targets than the ones trained with the initial database. The conditioning of the initial database turns out to be an essential step in obtaining an efficient trained neural network for solving the underlying statistical inverse problem. An *ad hoc* probabilistic model of the input random vector has been finally proposed in order to take into account experimental errors on the network input and to perform a robustness analysis of the network output with respect to the input uncertainties level. The challenging problem related to the identification of the input uncertainties level would deserve an in-depth analysis and should be part of a forthcoming work. The proposed neural-network based identification method has been successfully applied to

synthetic data and then carried out on real experimental data coming from experimental measurements on a beef cortical bone specimen. Although the proposed method has been developed for a simple 2D plane stress linear elasticity problem, it could be easily extended to more complicated 3D physical problems encountered in computational mechanics and engineering sciences. Finally, instead of using classical feedforward static (or series) neural networks, other neural network architectures may be considered to increase the network training speed and improve the neural network performance, such as multilayer cascade-forward neural networks (that may include additional feedforward connections), dynamic (or recurrent) neural networks (with feedback (or recurrent) connections and/or tapped delay lines) and directed acyclic graph (DAG) neural networks (that may include skipped layers or layers connected and operating in parallel), thus allowing for various neural network configurations and topologies to learn either static or dynamic (time-dependent) series relationships depending on the problem to be solved.

Acknowlegements

The authors gratefully acknowledge Christian Soize, Professor at Université Gustave Eiffel, Laboratoire MSME, for helpful discussions and valuable suggestions.

References

[1] C. Soize, Non-Gaussian positive-definite matrix-valued random fields for elliptic stochastic partial differential operators, Computer Methods in Applied Mechanics and Engineering 195 (1–3) (2006) 26–64. doi:10.1016/j.cma.2004.12.014.
URL https://doi.org/10.1016/j.cma.2004.12.014

[2] C. Soize, Tensor-valued random fields for meso-scale stochastic model of anisotropic elastic microstructure and probabilistic analysis of rep
Probabilistic Engineering Mechanics 23 (2–3) (2008) 307–323, 5th International Conference on Computational Stochastic Mechanics. doi:10.1016/j.probengmech.2007.12.019.
URL https://doi.org/10.1016/j.probengmech.2007.12.019

[3] M.-T. Nguyen, C. Desceliers, C. Soize, J.-M. Allain, H. Gharbi, Multiscale identification of the random elasticity field at mesoscale of a hete
International Journal for Multiscale Computational Engineering 13 (4) (2015) 281–295. doi:10.1615/IntJMultCompEng.2015011435.
URL https://doi.org/10.1615/IntJMultCompEng.2015011435

[4] T. Zhang, F. Pled, C. Desceliers, Robust Multiscale Identification of Apparent Elastic Properties at Mesoscale for Random Heterogeneous M
Materials 13 (12) (2020). doi:10.3390/ma13122826.
URL https://www.mdpi.com/1996-1944/13/12/2826

[5] C. Desceliers, R. Ghanem, C. Soize, Maximum likelihood estimation of stochastic chaos representations from experimental data, International Journal for Numerical Methods in Engineering 66 (6) (2006) 978–1001. doi:10.1002/nme.1576.
URL https://doi.org/10.1002/nme.1576

[6] R. G. Ghanem, A. Doostan, On the construction and analysis of stochastic models: Characterization and propagation of the errors associat
Journal of Computational Physics 217 (1) (2006) 63–81. doi:10.1016/j.jcp.2006.01.037.
URL https://doi.org/10.1016/j.jcp.2006.01.037

[7] C. Desceliers, C. Soize, R. Ghanem, Identification of Chaos Representations of Elastic Properties of Random Media Using Experimental Vi
Computational Mechanics 39 (6) (2007) 831–838. doi:10.1007/s00466-006-0072-7.
URL https://doi.org/10.1007/s00466-006-0072-7

[8] Youssef M. Marzouk and Habib N. Najm and Larry A. Rahn, Stochastic spectral methods for efficient bayesian solution of inverse problems,
Journal of Computational Physics 224 (2) (2007) 560–586. doi:10.1016/j.jcp.2006.10.010.
URL https://doi.org/10.1016/j.jcp.2006.10.010

[9] M. Arnst, D. Clouteau, M. Bonnet, Inversion of probabilistic structural models using measured transfer functions, Computer Methods in Applied Mechanics and Engineering 197 (6) (2008) 589–608. doi:10.1016/j.cma.2007.08.011.
URL https://doi.org/10.1016/j.cma.2007.08.011

[10] S. Das, R. Ghanem, J. C. Spall, Asymptotic Sampling Distribution for Polynomial Chaos Representation of Data: A Maximum Entropy and Fisher information approach, in: Proceedings of the 45th IEEE Conference on Decision and Control, 2006, pp. 4139–4144. doi:10.1109/CDC.2006.377613.

[11] S. Das, R. Ghanem, S. Finette, Polynomial chaos representation of spatio-temporal random fields from experimental measurements, Journal of Computational Physics 228 (23) (2009) 8726–8751. doi:10.1016/j.jcp.2009.08.025.
URL https://doi.org/10.1016/j.jcp.2009.08.025

[12] C. Desceliers, C. Soize, Q. Grimal, M. Talmant, S. Naili, Determination of the random anisotropic elasticity layer using transient wave propa
The Journal of the Acoustical Society of America 125 (4) (2009) 2027–2034. arXiv:https://doi.org/10.1121/1.3087428, doi:10.1121/1.3087428.
URL https://doi.org/10.1121/1.3087428

[13] J. Guilleminot, C. Soize, D. Kondo, Mesoscale probabilistic models for the elasticity tensor of fiber reinforced composites: Experimental ide Mechanics of Materials 41 (12) (2009) 1309–1322. doi:10.1016/j.mechmat.2009.08.004.
URL https://doi.org/10.1016/j.mechmat.2009.08.004

[14] X. Ma, N. Zabaras, An efficient Bayesian inference approach to inverse problems based on an adaptive sparse grid collocation method, Inverse Problems 25 (3) (2009) 035013.
URL http://stacks.iop.org/0266-5611/25/i=3/a=035013

[15] Y. M. Marzouk, H. N. Najm, Dimensionality reduction and polynomial chaos acceleration of Bayesian inference in inverse problems, Journal of Computational Physics 228 (6) (2009) 1862–1902. doi:10.1016/j.jcp.2008.11.024.
URL https://doi.org/10.1016/j.jcp.2008.11.024

[16] M. Arnst, R. Ghanem, C. Soize, Identification of Bayesian posteriors for coefficients of chaos expansions, Journal of Computational Physics 229 (9) (2010) 3134–3154. doi:10.1016/j.jcp.2009.12.033.
URL https://doi.org/10.1016/j.jcp.2009.12.033

[17] S. Das, J. C. Spall, R. Ghanem, Efficient Monte Carlo computation of Fisher information matrix using prior information, Computational Statistics & Data Analysis 54 (2) (2010) 272–289. doi:10.1016/j.csda.2009.09.018.
URL https://doi.org/10.1016/j.csda.2009.09.018

[18] Q.-A. Ta, D. Clouteau, R. Cottereau, Modeling of random anisotropic elastic media and impact on wave propagation, European Journal of Computational Mechanics 19 (1-3) (2010) 241–253. arXiv:http://www.tandfonline.com/doi/pdf/10.3166/ejcm.19.241-2 doi:10.3166/ejcm.19.241-253.
URL http://www.tandfonline.com/doi/abs/10.3166/ejcm.19.241-253

[19] C. Soize, Identification of high-dimension polynomial chaos expansions with random coefficients for non-Gaussian tensor-valued random fie Computer Methods in Applied Mechanics and Engineering 199 (33–36) (2010) 2150–2164. doi:10.1016/j.cma.2010.03.013.
URL https://doi.org/10.1016/j.cma.2010.03.013

[20] C. Soize, A computational inverse method for identification of non-Gaussian random fields using the Bayesian approach in very high dimer Computer Methods in Applied Mechanics and Engineering 200 (45–46) (2011) 3083–3099. doi:10.1016/j.cma.2011.07.005.
URL https://doi.org/10.1016/j.cma.2011.07.005

[21] R. Cottereau, D. Clouteau, H. B. Dhia, C. Zaccardi, A stochastic-deterministic coupling method for continuum mechanics, Computer Methods in Applied Mechanics and Engineering 200 (47-48) (2011) 3280–3288. doi:10.1016/j.cma.2011.07.010.
URL https://doi.org/10.1016/j.cma.2011.07.010

[22] C. Desceliers, C. Soize, S. Naili, G. Haiat, Probabilistic model of the human cortical bone with mechanical alterations in ultrasonic range, Mechanical Systems and Signal Processing 32 (2012) 170–177, uncertainties in Structural Dynamics. doi:10.1016/j.ymssp.2012.03.008.
URL https://doi.org/10.1016/j.ymssp.2012.03.008

[23] G. Perrin, C. Soize, D. Duhamel, C. Funfschilling, Identification of Polynomial Chaos Representations in High Dimension from a Set of Rea SIAM Journal on Scientific Computing 34 (6) (2012) A2917–A2945. arXiv:https://doi.org/10.1137/11084950X, doi:10.1137/11084950X.
URL https://doi.org/10.1137/11084950X

[24] D. Clouteau, R. Cottereau, G. Lombaert, Dynamics of structures coupled with elastic media—A review of numerical models and methods, Journal of Sound and Vibration 332 (10) (2013) 2415–2436. doi:10.1016/j.jsv.2012.10.011.
URL https://doi.org/10.1016/j.jsv.2012.10.011

[25] S. Haykin, Neural Networks: A Comprehensive Foundation, 1st Edition, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1994.

[26] M. T. Hagan, H. B. Demuth, M. H. Beale, Neural Network Design, PWS Publishing Co., Boston, MA, USA, 1996.

[27] H. B. Demuth, M. H. Beale, O. De Jess, M. T. Hagan, Neural Network Design, 2nd Edition, Martin Hagan, USA, 2014.
URL https://books.google.fr/books?id=4EW9oQEACAAJ

[28] A. W. Bowman, A. Azzalini, Applied Smoothing Techniques for Data Analysis, Oxford University Press, Oxford, 1997.

[29] I. Horová, J. Koláček, J. Zelinka, Kernel Smoothing in MATLAB: Theory and Practice of Kernel Smoothing, World Scientific Publishing Co. Pte. Ltd., Singapore, 2012.
URL http://www.worldscientific.com/worldscibooks/10.1142/8468#t=aboutBook

[30] G. H. Givens, J. A. Hoeting, Computational Statistics, 2nd Edition, John Wiley & Sons, Hoboken, New Jersey, 2013.

[31] D. W. Scott, Multivariate Density Estimation: Theory, Practice, and Visualization, 2nd Edition, John Wiley & Sons, Inc., 2015. doi:10.1002/9781118575574.
URL https://doi.org/10.1002/9781118575574

[32] C. Soize, Uncertainty Quantification: An Accelerated Course with Advanced Applications in Computational Engineering, 1st Edition, Vol. 47 of Interdisciplinary Applied Mathematics, Springer International Publishing, 2017. doi:10.1007/978-3-319-54339-0.
URL https://doi.org/10.1007/978-3-319-54339-0

[33] E. T. Jaynes, Information Theory and Statistical Mechanics, Phys. Rev. 106 (1957) 620–630. doi:10.1103/PhysRev.106.620.
URL http://link.aps.org/doi/10.1103/PhysRev.106.620

[34] E. T. Jaynes, Information Theory and Statistical Mechanics. II, Phys. Rev. 108 (1957) 171–190. doi:10.1103/PhysRev.108.171.
URL http://link.aps.org/doi/10.1103/PhysRev.108.171

[35] K. Sobezyk, J. Trębicki, Maximum entropy principle in stochastic dynamics, Probabilistic Engineering Mechanics 5 (3) (1990) 102–110. doi:10.1016/0266-8920(90)90001-Z.
URL https://doi.org/10.1016/0266-8920(90)90001-Z
[36] J. N. Kapur, H. K. Kesavan, Entropy Optimization Principles and Their Applications, Springer Netherlands, Dordrecht, 1992, pp. 3–20. doi:10.1007/978-94-011-2430-0_1.
URL https://doi.org/10.1007/978-94-011-2430-0_1
[37] G. Jumarie, Maximum Entropy, Information Without Probability and Complex Fractals: Classical and Quantum Approach, Vol. 112 of Fundamental Theories of Physics, Springer Science & Business Media, Dordrecht, 2000. doi:10.1007/978-94-015-9496-7.
URL https://doi.org/10.1007/978-94-015-9496-7
[38] E. T. Jaynes, Probability Theory: The Logic of Science, Cambridge university press, 2003.
[39] T. M. Cover, J. A. Thomas, Elements of Information Theory, A Wiley-Interscience publication, Wiley, New York, NY, USA, 2006.
[40] T. J. R. Hughes, The finite element method : linear static and dynamic finite element analysis, Prentice Hall, Englewood Cliffs, New Jersey, 1987.
[41] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, The Finite Element Method: Its Basis and Fundamentals, Butterworth-Heinemann, 2005.
[42] T. Zhang, Multiscale statistical inverse problem for the identification of random fields of elastic properties (in french), phdthesis, Université Paris-Est (Dec. 2019).
URL https://tel.archives-ouvertes.fr/tel-02506242
[43] S. Nemat-Nasser, M. Hori, Micromechanics: Overall Properties of Heterogeneous Materials, Vol. 37 of North-Holland Series in Applied Mathematics and Mechanics, North-Holland, Amsterdam, The Netherlands, 1993.
[44] M. Bornert, T. Bretheau, P. Gilormini, Homogénéisation en mécanique des matériaux 1. Matériaux aléatoires élastiques et milieux périodiques, Hermès Science publications, Paris, 2001.
[45] S. Torquato, Random Heterogeneous Materials: Microstructure and Macroscopic Properties, Vol. 16, Springer-Verlag, New York, NY, USA, 2002. doi:10.1007/978-1-4757-6355-3.
URL https://doi.org/10.1007/978-1-4757-6355-3
[46] A. Zaoui, Continuum Micromechanics: Survey, Journal of Engineering Mechanics 128 (8) (2002) 808–816. arXiv:https://ascelibrary.org/doi/pdf/10.1061/(ASCE)0733-9399(2002)128:8(808), doi:10.1061/(ASCE)0733-9399(2002)128:8(808).
URL https://doi.org/10.1061/(ASCE)0733-9399(2002)128:8(808)
[47] A. Bourgeat, A. Piatnitski, Approximations of effective coefficients in stochastic homogenization, Annales de l'Institut Henri Poincare (B) Probability and Statistics 40 (2) (2004) 153–165. doi:10.1016/j.anihpb.2003.07.003.
URL https://doi.org/10.1016/j.anihpb.2003.07.003
[48] M.-T. Nguyen, J.-M. Allain, H. Gharbi, C. Desceliers, C. Soize, Experimental multiscale measurements for the mechanical identification of a
Journal of the Mechanical Behavior of Biomedical Materials 63 (2016) 125–133. doi:10.1016/j.jmbbm.2016.06.011.
URL https://doi.org/10.1016/j.jmbbm.2016.06.011
[49] M. Shinozuka, Simulation of Multivariate and Multidimensional Random Processes, The Journal of the Acoustical Society of America 49 (1B) (1971) 357–368. arXiv:https://doi.org/10.1121/1.1912338, doi:10.1121/1.1912338.
URL https://doi.org/10.1121/1.1912338
[50] M. Shinozuka, Y. K. Wen, Monte Carlo Solution of Nonlinear Vibrations, AIAA Journal 10 (1) (1972) 37–40. doi:10.2514/3.50064.
URL https://doi.org/10.2514/3.50064
[51] M. Shinozuka, C.-M. Jan, Digital simulation of random processes and its applications, Journal of Sound and Vibration 25 (1) (1972) 111–128. doi:10.1016/0022-460X(72)90600-1.
URL https://doi.org/10.1016/0022-460X(72)90600-1
[52] F. Poirion, C. Soize, Numerical simulation of homogeneous and inhomogeneous Gaussian stochastic vector fields, La Recherche Aerospatiale (English edition) 1 (-) (1989) 41–61.
URL https://hal-upec-upem.archives-ouvertes.fr/hal-00770316
[53] F. Poirion, C. Soize, Numerical methods and mathematical aspects for simulation of homogeneous and non homogeneous gaussian vector fi
in: P. Krée, W. Wedig (Eds.), Probabilistic Methods in Applied Physics, Springer Berlin Heidelberg, Berlin, Heidelberg, 1995, pp. 17–53. doi:10.1007/3-540-60214-3_50.
URL https://doi.org/10.1007/3-540-60214-3_50
[54] B. W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman and Hall, London, 1986.
[55] C. Robert, G. Casella, Monte Carlo Statistical Methods, 2nd Edition, Springer Texts in Statistics, Springer-Verlag, New York, NY, USA, 2004. doi:10.1007/978-1-4757-4145-2.
URL https://dx.doi.org/10.1007/978-1-4757-4145-2
[56] J. Kaipio, E. Somersalo, Statistical and Computational Inverse Problems, 1st Edition, Vol. 160 of Applied Mathematical Sciences, Springer-Verlag, New York, NY, USA, 2005. doi:10.1007/b138659.
URL https://doi.org/10.1007/b138659
[57] J. C. Spall, Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control, Vol. 65, John Wiley & Sons, 2005. doi:10.1002/0471722138.
URL https://dx.doi.rg/10.1002/0471722138
[58] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals and Systems 2 (4) (1989) 303–314. doi:10.1007/BF02551274.

URL https://doi.org/10.1007/BF02551274
[59] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366. doi:10.1016/0893-6080(89)90020-8.
URL https://doi.org/10.1016/0893-6080(89)90020-8
[60] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks 4 (2) (1991) 251–257. doi:10.1016/0893-6080(91)90009-T.
URL https://doi.org/10.1016/0893-6080(91)90009-T
[61] M. Leshno, V. Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate a
Neural Networks 6 (6) (1993) 861–867. doi:10.1016/S0893-6080(05)80131-5.
URL https://doi.org/10.1016/S0893-6080(05)80131-5
[62] A. R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Transactions on Information Theory 39 (3) (1993) 930–945. doi:10.1109/18.256500.
URL https://doi.org/10.1109/18.256500
[63] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444. doi:10.1038/nature14539.
URL https://doi.org/10.1038/nature14539
[64] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, Cambridge, MA, USA, 2016, http://www.deeplearningbook.org.
[65] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, D. L. Alkon, Accelerating the convergence of the back-propagation method, Biological Cybernetics 59 (4) (1988) 257–263. doi:10.1007/BF00332914.
URL https://doi.org/10.1007/BF00332914
[66] D. Nguyen, B. Widrow, Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights, in: 1990 IJCNN International Joint Conference on Neural Networks, Vol. 3, 1990, pp. 21–26. doi:10.1109/IJCNN.1990.137819.
URL https://doi.org/10.1109/IJCNN.1990.137819
[67] M. H. Beale, M. T. Hagan, H. B. Demuth, Neural network toolbox user's guide, The MathWorks Inc (1992).
[68] C. Soize, Random matrix theory for modeling uncertainties in computational mechanics, Computer Methods in Applied Mechanics and Engineering 194 (12–16) (2005) 1333–1366, special Issue on Computational Methods in Stochastic Mechanics and Reliability Analysis. doi:10.1016/j.cma.2004.06.038.
URL https://doi.org/10.1016/j.cma.2004.06.038
[69] C. Soize, Random Matrix Models and Nonparametric Method for Uncertainty Quantification, Springer International Publishing, Cham, 2016, pp. 1–69. doi:10.1007/978-3-319-11259-6_5-1.
URL https://doi.org/10.1007/978-3-319-11259-6_5-1
[70] C. Lawson, R. Hanson, Solving Least Squares Problems, Society for Industrial and Applied Mathematics, 1995. arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611971217, doi:10.1137/1.9781611971217.
URL https://doi.org/10.1137/1.9781611971217
[71] R. Serfling, Approximation Theorems of Mathematical Statistics, Wiley, New York, NY, USA, 1980. doi:10.1002/9780470316481.
URL https://dx.doi.org/10.1002/9780470316481
[72] A. Papoulis, S. U. Pillai, Probability, Random Variables, and Stochastic Processes, 4th Edition, McGraw-Hill Higher Education, New York, NY, USA, 2002.